Enumerating edge-constrained triangulations and edge-constrained non-crossing geometric spanning trees

AUTHOR(S):

Katoh, Naoki; Tanigawa, Shin-ichi

CITATION:

Katoh, Naoki ...[et al]. Enumerating edge-constrained triangulations and edge-constrained non-crossing geometric spanning trees. Discrete Applied Mathematics 2009, 157(17): 3569-3585

ISSUE DATE:

2009-10-28

URL:

http://hdl.handle.net/2433/123380

Enumerating Edge-constrained Triangulations and Edge-constrained Non-crossing Geometric Spanning Trees

Naoki Katoh and Shin-ichi Tanigawa

October 10, 2008

Abstract

In this paper we present algorithms for enumerating without repetitions all triangulations and non-crossing geometric spanning trees on a given set of n points in the plane under edge inclusion constraint (i.e., some edges are required to be included in the graph). We will first extend the lexicographically ordered triangulations introduced by Bespamyatnikh to the edge-constrained case, and then we prove that a set of all edge-constrained non-crossing spanning trees is connected via remove-add flips, based on the edge-constrained lexicographically largest triangulation. More specifically, we prove that all edge-constrained triangulations can be transformed to the lexicographically largest triangulation among them by $O(n^2)$ greedy flips, i.e., by greedily increasing the lexicographical ordering of the edge list, and a similar result also holds for a set of edge-constrained non-crossing spanning trees. Our enumeration algorithms generate each output triangulation and non-crossing spanning tree in $O(\log \log n)$ and $O(n^2)$ time, respectively, based on the reverse search technique.

Keywords: geometric enumeration; edge-constrained triangulations; edge-constrained non-crossing spanning trees.

1 Introduction

Given a graph $G = (V, E)$ with n vertices and m edges where $V = \{1, \ldots, n\}$. An embedding of the graph on a set of points $P = \{p_1, \cdots, p_n\} \subset \mathbf{R}^2$ is a mapping of $i \in V$ to $p_i \in P$. A *geometric graph (on P)* is a graph embedded on P such that each edge (i, j) of G is mapped to the straight line segment (p_i, p_j). The point set P is assumed to be fixed in $\mathbf{R}^2$, and n denotes the cardinality of P throughout the paper. The geometric graph is *non-crossing* if each pair of segments (p_i, p_j) and (p_k, p_l) have no point in common without their endpoints. Similarly, a set of line segments is called non-crossing if any pair of line segments have a point in common without their endpoints. A set of line segments is *on* P if all endpoints of the segments are points of P. For a set F of non-crossing line segments on P, a non-crossing geometric graph containing F is called an *F-constrained non-crossing geometric graph.*

In this paper we shall provide algorithms for enumerating all the *F-constrained triangulations* and the *F-constrained non-crossing spanning trees* (embedded) on P. The proposed algorithm of F-constrained triangulations requires $O(\log \log n)$ time per output triangulation. This is a direct extension of the algorithm for enumerating (unconstrained) triangulations by Bespamyatnikh [12].

[1]Department of Architecture and Architectural Engineering, Kyoto Daigaku Katsura, Nishikyo-ku, Kyoto 615-8540 Japan, {naoki,is.tanigawa}@archi.kyoto-u.ac.jp.

[2]A preliminary version of this paper has appeared in the Proceedings of COCOON 2007, Lecture Notes in Computer Science 4598, pages 243–253, Springer Verlag.

The proposed algorithm of F-constrained non-crossing spanning trees requires $O(n^2)$ time per output using $O(n^2)$ space. For the unconstrained case (i.e. $F = \emptyset$), the algorithm by Avis and Fukuda [8] requires $O(n^3)$ time per output and $O(n)$ space. Recently, Aichholzer et al. [2] have developed an algorithm for enumerating all non-crossing spanning trees in $O(n \log n)$ time per output based on the Gray code enumeration, whose space complexity is not given. Although the algorithm of [2] is superior to ours in the unconstrained case, it seems that it cannot be extended to the edge-constrained case. In particular, it is not trivial to show that the collection of all the F-constrained non-crossing spanning trees is connected via a removed-add flip operation.

It is well known that the number of the triangulations or the non-crossing spanning trees grows too rapidly to allow a complete enumeration on a significantly large point set (see e.g. [4]). In view of practical applications the number of objects to be enumerated or the computational cost should be reduced by imposing several reasonable constraints. For this purpose, the edge constraint would be naturally considered.

For the edge-constrained case, in our recent paper [9], we proposed an algorithm for enumerating the edge-constrained non-crossing minimally rigid frameworks embedded on a given point set in the plane in $O(n^3)$ time per output graph. We remarked therein that based on a similar approach, we could develop an $O(n^3)$ time algorithm for enumerating edge-constrained non-crossing spanning trees. Although we have not given either any algorithmic details or the analysis of the running time, it seems difficult to improve this running time.

Let $\mathcal{O}$ be the set of graphs to be enumerated. Two graphs are *connected* if and only if they can be transformed to each other by a *local operation*, which generates one graph from the other by means of a small change. In particular, it is often called a *(*1*-)flip* if it removes an edge from the graph and then inserts the other edge to obtain a new graph. Define the graph $\mathcal{G}_{\mathcal{O}}$ on $\mathcal{O}$ with the set of edges connecting two graphs of $\mathcal{O}$ that can be transformed to each other by a specified local operation. Then, the natural question is how we can design a local operation so that $\mathcal{G}_{\mathcal{O}}$ is connected, or how we can design $\mathcal{G}_{\mathcal{O}}$ with the small diameter. There are several known results for these questions for triangulations (e.g. [18]), pseudo-triangulations [1, 10, 13], geometric matchings [16, 17], some classes of simple polygons [15] and also for non-crossing spanning trees [1–3, 5, 8].

It is well known that every triangulation on a fixed point set can be transformed to Delaunay triangulation by $O(n^2)$ diagonal flips, and this result can be naturally extended to the edge-constrained triangulations (see e.g. [11]). Bespamyatnikh [12] showed the other sequence of the diagonal flips to develop an efficient algorithm for enumerating triangulations, where he focused on the lexicographically ordered edge list of each triangulations and showed that every triangulation can be transformed into the one having the lexicographically largest edge list by $O(n^2)$ greedy flips. We will extend this result to the edge-constrained triangulations.

As for the collection $\mathcal{ST}$ of the non-crossing spanning trees on P, Avis and Fukuda [8] have developed a 1-flip such that $\mathcal{G}_{\mathcal{ST}}$ is connected with diameter $2n-4$. Aichholzer et al. in [2] showed that $\mathcal{G}_{\mathcal{ST}}$ defined by a 1-flip contains a Hamiltonian path, which provides a Gray code enumeration scheme. Aichholzer et al. in [3, 5] tried to design a 1-flip with the additional requirement, called *edge slide*, such that the removed edge moves to the other one along an adjacent edge keeping one endpoint of the removed edge fixed. In this paper we will propose a 1-flip that increases the lexicographical ordering of the edge list of the (edge-constrained) non-crossing spanning trees and show that every (edge-constrained) non-crossing spanning tree can be transformed to one particular non-crossing spanning tree that has the lexicographically largest edge list by $O(n^2)$ flips. We remark that it seems difficult to extend all the 1-flips designed in the previous works to the edge-constrained case. We also remark that, for the case of an operation other than a 1-flip, which removes and inserts more than one edge preserving some specified rules, the operations with

diameters of $O(\log n)$ [3] and the improved result [1] are known.

A main tool we use in our enumeration algorithms is the reverse search technique developed by Avis and Fukuda [7, 8]. The reverse search generates all the elements of $\mathcal{O}$ by tracing the nodes in $\mathcal{G}_{\mathcal{O}}$. To trace $\mathcal{G}_{\mathcal{O}}$ efficiently, it defines the *root* on $\mathcal{G}_{\mathcal{O}}$ and the *parent* for each node except for the root. Define the parent-child relation that satisfies the following conditions: (1) each non-root object has the unique parent, and (2) an ancestor of an object is not itself. Then, iterating going up to the parent leads to the root from any other node in $\mathcal{G}_{\mathcal{O}}$ if $\mathcal{G}_{\mathcal{O}}$ is *connected*. The collection of these paths induces a spanning tree, known as a *search tree*, and the algorithm traces it by depth-first manner. Hence, the necessary ingredients to use the method are an implicitly described connected graph $\mathcal{G}_{\mathcal{O}}$ and an implicitly defined search tree on $\mathcal{G}_{\mathcal{O}}$. In this paper we supply these ingredients for the problems of generating all the F-constrained triangulations and the F-constrained non-crossing spanning trees on P.

2 Lexicographically Ordered Edge-constrained Triangulation

In this section we introduce the *F-constrained lexicographically largest triangulation* (F-CLLT) on P, and then we show that every F-constrained triangulation can be transformed into the F-CLLT by $O(n^2)$ flips. We remark again that F-CLLT is derived from the lexicographically ordered triangulation developed by Bespamyatnikh [12] although he did not extend his result to the edge-constrained case.

2.1 Notations

We assume that x-coordinates of all points of P are distinct and no three points of P are colinear. We label the points of P as $p_1, \ldots, p_n$ in the increasing order of x-coordinates. For two vertices $p_i, p_j \in P$, we denote $p_i < p_j$ if $i < j$ holds. Considering $p_i \in P$, we often pay our attention only to its right point set, $\{p_{i+1}, \ldots, p_n\} \subseteq P$, which is denote by P_{i+1}.

Let K_n be the complete graph embedded on P (with the straight line segments), and the line segment between p_i and p_j with $p_i < p_j$ is called an *edge* between p_i and p_j, denoted by (p_i, p_j). We often use the notation G to denote the edge set of a (geometric) graph G for simplicity when it is clear from the context.

For three points p_i, p_j and p_k the signed area $\Delta(p_i, p_j, p_k)$ of the triangle (p_i, p_j, p_k) tells us that p_k is on the left or right side of the line passing through p_i and p_j when moving along the line from p_i to p_j by $\Delta(p_i, p_j, p_k) > 0$ or $\Delta(p_i, p_j, p_k) < 0$, respectively. A *total ordering* $\prec$ on a set of edges is defined as follows: for $e = (p_i, p_j)$ and $e' = (p_k, p_l)$ (with $p_i < p_j$ and $p_k < p_l$), e is *smaller* than e', denoted by $e \prec e'$, if and only if $p_i < p_k$, or $p_i = p_k$ and $\Delta(p_i, p_j, p_l) < 0$. Note that, when $p_i = p_k$, this ordering corresponds to the clockwise ordering around p_i. Let $E = \{e_1, \ldots, e_m\}$ and $E' = \{e'_1, \ldots, e'_m\}$ be two sorted edge lists with $e_1 \prec \cdots \prec e_m$ and $e'_1 \prec \cdots \prec e'_m$. Then, E' is lexicographically larger than E if $e_i \prec e'_i$ for the smallest i such that $e_i \neq e'_i$.

We say that two edges (p_i, p_j) and (p_k, p_l) *properly intersect* each other if (p_i, p_j) and (p_k, p_l) have a point in common except for their endpoints. Let F be a non-crossing edge set on P. For two points $p_i, p_j \in P$, p_j is *visible* from p_i with respect to F when (p_i, p_j) properly intersects no edge of F. We assume that p_j is visible from p_i if $(p_i, p_j) \in F$.

The *upper tangent* (p_i, p_i^{up}) and the *lower tangent* $(p_i, p_i^{\mathrm{low}})$ *of p_i with respect to F* are defined as those from p_i to the convex hull of the points of P_{i+1} that are visible from p_i with respect to F (see Fig. 1). Notice that each of the upper and lower tangents defines an *empty region* in which no point of P exists as described below. Let l be the line perpendicular to the x-axis passing through p_i, and let e_1 and e_2 be the closest edges from p_i among F intersecting with l in the upper and

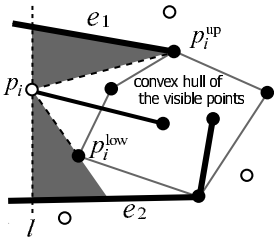


Figure 1: An example of the upper and lower tangents, denoted by (p_i, p_i^{up}) and $(p_i, p_i^{\mathrm{low}})$, respectively. The bold edges represent F. The empty regions of (p_i, p_i^{up}) and $(p_i, p_i^{\mathrm{low}})$ are shaded.

lower sides of p_i, respectively (if such edge exists). Then there exists no point of P inside the region bounded by l, e_1 (resp. e_2) and the line passing through p_i and p_i^{up} (resp. p_i^{low}). When e_1 (resp. e_2) does not exist, the empty region is defined by the one bounded by l and the line through p_i and p_i^{up} (resp. p_i^{low}). Thus, we have the following fact:

Observation 2.1. *Let F be a non-crossing edge set on P. Then, for any edge $e \in K_n$ that properly intersects either the upper or lower tangent of p_i with respect to F, at least one of the following two facts holds:* (1) *the left endpoint of e is less than p_i and* (2) *e properly intersects some edge of F.*

2.2 Edge-constrained Lexicographically Largest Triangulation

For $p_i \in P$ and a geometric graph G on P, let us denote by $\delta_G(p_i)$ the set of edges of G which are incident to p_i with the left endpoints. Similarly, for an edge set F on P, $\delta_F(p_i)$ denotes the set of edges of F which are incident to p_i with the left endpoints. Let us consider the following construction of the F-constrained geometric graph on P for a non-crossing edge set F:

Construction 1.

0. *Repeat the following process for all $p_i \in P$ in an arbitrary order.*

1. *Let (p_i, p_i^{up}) and $(p_i, p_i^{\mathrm{low}})$ be the upper and lower tangents of $p_i \in P$ with respect to F, and denote the right endpoints of $\delta_F(p_i) \cup \{(p_i, p_i^{\mathrm{up}}), (p_i, p_i^{\mathrm{low}})\}$ by $p_{i_0}, p_{i_1}, \ldots, p_{i_m}$ in clockwise order around p_i (where $p_{i_0} = p_i^{\mathrm{up}}$ and $p_{i_m} = p_i^{\mathrm{low}}$ hold) (Fig.2(a)).*

2. *Consider the cone C_k with the apex p_i bounded by two consecutive edges (p_i, p_{i_k}) and $(p_i, p_{i_{k+1}})$ for each k with $0 \leq k \leq m-1$, where C_k contains both p_{i_k} and $p_{i_{k+1}}$, and construct the convex hull H_k of $P_{i+1} \cap C_k$ inside each C_k (Fig.2(b)).*

3. *Connect from p_i to every point $p_j \in P_{i+1} \cap C_k$ if $p_j = (p_i, p_j) \cap H_k$ holds for some k (Fig.2(c)).*

We give an example of the graph obtained by Construction 1 in Fig. 3. Notice that the graph obtained by Construction 1 always has the edges of $\delta_F(p_i) \cup \{(p_i, p_i^{\mathrm{up}}), (p_i, p_i^{\mathrm{low}})\}$ for all $p_i \in P$. In addition, the following property could be easily observed:

Lemma 2.2. *Let F be a non-crossing edge set on a given point set P. Let G be the graph obtained by Construction 1 for F and let (p_i, p_j) be an edge of G. Then, any edge of K_n properly intersecting (p_i, p_j) also properly intersects at least one edge of $\delta_F(p_i) \cup \{(p_i, p_i^{\mathrm{up}}), (p_i, p_i^{\mathrm{low}})\}$.*

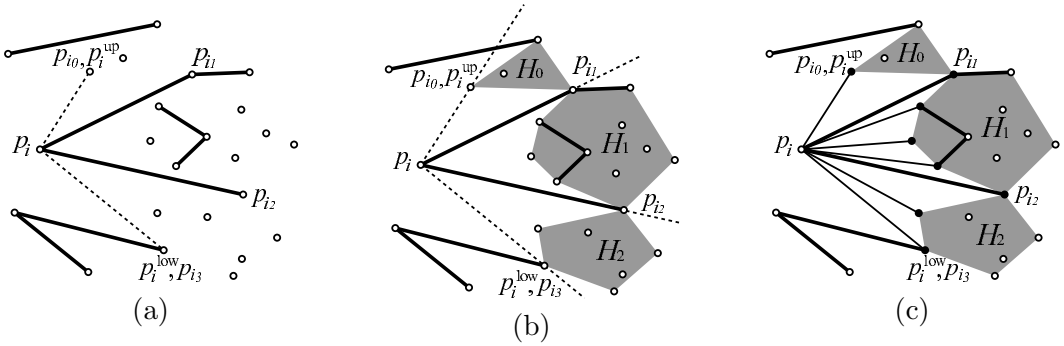


Figure 2: Construction 1 around p_i, where bold edges represent F. (a) Step 1, (b) Step 2 and (c) Step 3.

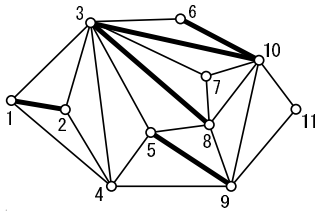


Figure 3: CLLT.

Proof. Let us consider Construction 1 around p_i. Then, there exists a convex hull H_k for which $p_j = (p_i, p_j) \cap H_k$ from the definition of Construction 1. Notice that the two consecutive edges, (p_i, p_{i_k}) and $(p_i, p_{i_{k+1}})$ of $\delta_F(p_i) \cup \{(p_i, p_i^{\text{up}}), (p_i, p_i^{\text{low}})\}$ (bounding C_k considered in Step 2), and the part of the boundary of H_k from p_{i_k} to $p_{i_{k+1}}$ (that is a convex chain) forms a simple polygon with exactly three convex vertices, p_i, p_{i_k} and $p_{i_{k+1}}$, which is a so-called *pseudo-triangle*. Recall that p_j is a vertex of the pseudo-triangle because $p_j = (p_i, p_j) \cap H_k$. Since there exists no point of P inside of the pseudo-triangle, any edge properly intersecting (p_i, p_j) must properly intersect at least one of (p_i, p_{i_k}) and $(p_i, p_{i_{k+1}})$. □

The following lemmas describe the fundamental properties of the above defined construction:

Lemma 2.3. *The graph G obtained by Construction 1 is a triangulation on P.*

Proof. We will prove, by induction on i from $i = n$ to 1, that (1) the subgraph of G induced by P_i, denoted by G_i, is non-crossing, and (2) all faces of G_i are triangles except for the outer face. This implies that G is a triangulation since G clearly contains the boundary edges of the convex hull of P from the definition of Construction 1.

For the basis, G_n has no edge, and hence the statement holds. Assume that (1) and (2) hold for G_{i+1}. We first show that (1) holds for G_i. Suppose there exists an edge $(p_a, p_b) \in G_{i+1}$ with $p_a < p_b$ that properly intersects some edge of $G_i \setminus G_{i+1}$. Then, from Lemma 2.2, (p_a, p_b) properly intersects some edge of $\delta_F(p_i) \cup \{(p_i, p_i^{\text{up}}), (p_i, p_i^{\text{low}})\}$. By Construction 1 it is obvious that (p_a, p_b) does not properly intersect any edge of F. Hence (p_a, p_b) properly intersects either (p_i, p_i^{up}) or (p_i, p_i^{low}). However, this implies, by Observation 2.1, that p_a lies on the left side of p_i, which contradicts $p_a \in P_{i+1}$.

Let us prove (2). Let (p_i, p_a) and (p_i, p_b) be two consecutive edges of $G_i \setminus G_{i+1}$ in clockwise order around p_i. From the definition of Construction 1, there exists the convex hull H_k such that p_a and p_b are consecutive vertices on the boundary of H_k. Hence, an edge between p_a and p_b is one of the upper or lower tangents of p_a or p_b with respect to F, and thus it is contained in G_{i+1}

by Construction 1. Moreover, from the definition of H_k given in Construction 1, the triangle face of (p_i, p_a, p_b) contains no point of P, and thus (2) follows. As a result, G is an F-constrained triangulation on P. □

Lemma 2.4. *The F-constrained triangulation $T^*(F)$ obtained by Construction 1 has the lexicographically largest edge list among all the F-constrained triangulations on P.*

Proof. Let us denote the edges of $T^*(F)$ by $\{e_1^*, \ldots, e_m^*\}$ with $e_1^* \prec \cdots \prec e_m^*$. Suppose there exists an F-constrained triangulation T whose edge set $\{e_1, \ldots, e_m\}$ with $e_1 \prec \cdots \prec e_m$ is lexicographically larger than that of $T^*(F)$. Then, there exists the smallest label s with $e_s^* \neq e_s$ for which $e_s^* \notin T$ and $e_s^* \prec e_s$ hold.

Let $e_s^* = (p_i, p_j) \in T^*(F) \setminus T$. Since s is the smallest label among the edges e_i for which $e_i^* \neq e_i$, $\delta_{T^*(F)}(p) = \delta_T(p)$ holds for every $p \in \{p_1, \ldots, p_{i-1}\}$. Since T is a triangulation but does not contain e_s^*, T must contain at least one edge $e \notin T^*(F)$ that properly intersects e_s^*. By Lemma 2.2, e properly intersects some edge of $\delta_F(p_i) \cup \{(p_i, p_i^{\mathrm{up}}), (p_i, p_i^{\mathrm{low}})\}$. In addition, since T is an F-constrained triangulation, e does not properly intersect any edge of $\delta_F(p_i)$, and consequently e properly intersects at least (p_i, p_i^{up}) or $(p_i, p_i^{\mathrm{low}})$. Observation 2.1 hence implies that the left endpoint of e is on the left side of p_i, which contradicts $\delta_{T^*(F)}(p) = \delta_T(p)$ for $p \in \{p_1, \ldots, p_{i-1}\}$. □

Hence, we call the F-constrained triangulation obtained by the above construction the *F-constrained lexicographically largest triangulation* (F-CLLT). In fact we can show that F-CLLT can be constructed by the greedily adding the edges to F in the descending edge ordering without violating the non-crossing property.

2.3 Improving Flips

Let $T^*(F)$ denote the F-CLLT on P. For any F-constrained triangulation T with $T \neq T^*(F)$, the *critical vertex* of T is the vertex having the smallest label among those incident to some edge in $T \setminus T^*(F)$. For two F-constrained triangulations T and T', T is called lexicographically larger than T' when the edge list of T is lexicographically larger than that of T'.

For an edge e with $e \in T \setminus F$, e is called *flippable* if two triangles incident to e in T form a *convex* quadrilateral Q. *Flipping* e in T generates a new F-constrained triangulation by replacing e with the other diagonal of Q. Such an operation is called an *improving flip* if the triangulation obtained by flipping e is lexicographically larger than the previous one, and e is called *improving flippable*. Note that we are playing on the collection of the *F-constrained* triangulations for given P and F, and thus it is assumed that the edges of F cannot be flippable. Now let us show a sequence of the improving flips.

Lemma 2.5. *Let T be an F-constrained triangulation with $T \neq T^*(F)$ and p_c be the critical vertex of T. Then, there exists at least one improving flippable edge incident to p_c in $T \setminus T^*(F)$.*

Proof. Let (p_c, p_c^{up}) and $(p_c, p_c^{\mathrm{low}})$ be the upper and lower tangents of p_c with respect to F. We shall first show $\delta_{T^*(F)}(p_c) \subset T$.

It is obvious that T contains every edge of $\delta_F(p_c)$ because T is an F-constrained triangulation. Let us show that $(p_c, p_c^{\mathrm{up}}) \in T$. Suppose (p_c, p_c^{up}) is missing in T. Then, T has some edge $(p_a, p_b) \notin T^*(F)$ that properly intersects (p_c, p_c^{up}) since T is a triangulation. Moreover, by Observation 2.1, $p_a < p_c$ holds, implying that p_a is incident to $(p_a, p_b) \notin T^*(F)$ and contradicting that p_c is the critical vertex of T. Thus, $(p_c, p_c^{\mathrm{up}}) \in T$ holds and also the same argument can be applied to $(p_c, p_c^{\mathrm{low}}) \in T$.

Next let us show that every edge $(p_c, p) \in \delta_{T^*(F)}(p_c)$ other than $\delta_F(p_c) \cup \{(p_c, p_c^{\mathrm{up}}), (p_c, p_c^{\mathrm{low}})\}$ is contained in T. Suppose (p_c, p) is missing in T. Then, there exists some edge $e \in T \setminus T^*(F)$

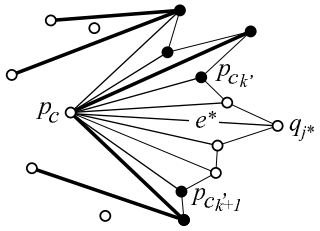


Figure 4: Existence of an improving flippable edge $e^* = (p_c, q_{j^*})$. The bold edges represent the edges of F, and the black vertices represent the vertices incident to p_c in $T^*(F)$.

that properly intersects (p_c, p). Lemma 2.2 now implies that e also properly intersects some edge of $\delta_F(p_c) \cup \{(p_c, p_c^{\mathrm{up}}), (p_c, p_c^{\mathrm{low}})\}$, contradicting that T contains all the edges of $\delta_F(p_c) \cup \{(p_c, p_c^{\mathrm{up}}), (p_c, p_c^{\mathrm{low}})\}$. Therefore, we have $\delta_{T^*(F)}(p_c) \subset T$.

Now let us show that there exists at least one improving flippable edge incident to p_c. Since p_c is the critical vertex, there exists an edge e in T incident to p_c with $e \notin T^*(F)$. Let $(p_c, p_{c_{k'}})$ and $(p_c, p_{c_{k'+1}})$ be two consecutive edges of $\delta_{T^*(F)}(p_c)(\subset T)$ around p_c such that e exists between $(p_c, p_{c_{k'}})$ and $(p_c, p_{c_{k'+1}})$ (see Fig. 4). Consider the edge subset of T incident to p_c between $(p_c, p_{c_{k'}})$ and $(p_c, p_{c_{k'+1}})$, and denote the elements of the subset by $(p_c, q_0), (p_c, q_1), \ldots, (p_c, q_l)$ in clockwise order around p_c, where $q_0 = p_{c_{k'}}$ and $q_l = p_{c_{k'+1}}$. Then, $(p_c, q_j) \in T \setminus T^*(F)$ holds for all $j = 1, \ldots, l-1$, and moreover any of $q_1, q_2, \ldots, q_{l-1}$ is not inside of the triangle $(p_c, p_{c_{k'}}, p_{c_{k'+1}})$ since $T^*(F)$ has the empty triangle face $(p_c, p_{c_{k'}}, p_{c_{k'+1}})$. Therefore, every (p_c, q_j) properly intersects the line segment connecting $p_{c_{k'}}$ and $p_{c_{k'+1}}$. Let q_{j^*} be the vertex furthest from the line passing through $p_{c_{k'}}$ and $p_{c_{k'+1}}$ among q_j. Then, the quadrilateral $p_c q_{j^*-1} q_{j^*} q_{j^*+1}$ is convex because q_{j^*-1}, q_{j^*} and q_{j^*+1} are not colinear, and flipping $e^* = (p_c, q_{j^*})$ produces a lexicographically larger triangulation than T because $p_c < q_{j^*-1}$ and $p_c < q_{j^*+1}$ hold. □

Theorem 2.6. *Let P be a set of n points in the plane. Every F-constrained triangulation T on P can be transformed to the F-CLLT on P by $O(n^2)$ improving flips.*

Proof. From Lemma 2.5, $T(\neq T^*(F))$ always has an improving flippable edge, and flipping such edge reduces the number of edges of $T \setminus T^*(F)$ incident to the critical vertex p_c. Moreover, the improving flip never decreases the label of the critical vertex. Hence, after $O(n)$ improving flips, the label of the critical vertex increases by at least one. Therefore, T can be transformed to the F-CLLT by $O(n^2)$ improving flips. □

The rest of this section describes the enumeration of the F-constrained triangulations on P. As we have proved that the lexicographical order of the (unconstrained) triangulations can be naturally extended to the edge-constrained case above, the algorithm for the unconstrained case by Bespamyatnikh [12] that is based on the lexicographical order of the unconstrained triangulations can be also extended to the edge-constrained case. For every F-constrained triangulation T with $T \neq T^*(F)$, let us define the *parent* of T as the triangulation obtained by flipping the smallest improving flippable edge among $T \setminus T^*(F)$ with respect to the edge ordering $\prec$, which surely exists from Lemma 2.5. Then, from the correctness of Theorem 2.6, these parent-child relations form the *search tree* of the triangulations on P explained in Introduction (whose root is $T^*(F)$).

It is known that the time complexity of the reverse search relies on the efficiency of finding the children of each object; in our case finding the children of each F-constrained triangulation. This task can be done by using the algorithm for the unconstrained case by just ignoring the edges of F in the algorithm by Bespamyatnikh [12] and thus we can obtain the algorithm that works in the

same time complexity as that of the unconstrained case (see Section 4 of [12]). Thus, we obtain the following result:

Theorem 2.7. *Let P be a set of n points in the plane. Then, all the F-constrained triangulations on P can be reported in $O(\log \log n)$ time per output graph with linear space.*

3 Deleting and Inserting the Constrained Edges

In this section we will discuss how the edge-constrained lexicographically largest triangulation changes when removing a constrained edge or inserting a new one. In order to describe the properties of Construction 1 in the general form, we shall use the notation E to denote a non-crossing edge set on P (rather than F, which is used to denote a given (fixed) edge-constraint throughout the paper), and we shall utilize Construction 1 as a function T^* that maps a non-crossing edge set E to the corresponding E-constrained lexicographically largest triangulation $T^*(E)$. The following facts will be heavily used mainly in Section 6 to develop an efficient enumeration algorithm for F-constrained non-crossing spanning trees. Let us first consider the case in which we insert a new constrained edge e to $T^*(E)$.

Lemma 3.1. *Let E be a non-crossing edge set, and let e be an edge of K_n that properly intersects no edge of E. Let I_e be the set of edges of $T^*(E)$ that properly intersect e. Then, $T^*(E+e)$ contains all the edges of $T^*(E) \setminus I_e$.*

Proof. Let $(p_i, p_j) \in T^*(E) \setminus I_e$. Note that p_j is still visible from p_i with respect to $E+e$. Consider two cones, C_E and C_{E+e}, obtained in Construction 1 for $T^*(E)$ and $T^*(E+e)$, respectively, with the apex p_i and containing p_j. Let H_E and H_{E+e} be the convex hulls of $P_{i+1} \cap C_E$ and $P_{i+1} \cap C_{E+e}$, (each of which contains p_j). When inserting e, the vertices that are not visible from p_i with respect to E remain non-visible from p_i with respect to $E+e$ although some of vertices visible from p_i with respect to E may become non-visible from p_i with respect to $E+e$. This implies $H_{E+e} \subseteq H_E$. Moreover, $(p_i, p_j) \in T^*(E)$ implies $p_j = (p_i, p_j) \cap H_E$ by the definition of Construction 1. Thus, we have $p_j = (p_i, p_j) \cap H_{E+e}$ and (p_i, p_j) remains in $T^*(E+e)$. □

As a corollary we obtain the following fact:

Lemma 3.2. *Let E be a non-crossing edge set. For every $e \in T^*(E)$, $T^*(E+e) = T^*(E)$ holds.*

Next let us consider the case in which we remove a constrained edge $e \in E$ from $T^*(E)$.

Lemma 3.3. *Let E be a non-crossing edge set. Then, for $e = (p_i, p_j) \in E$, $T^*(E-e) = T^*(E)$ holds if either*

(i) *e is either the upper or lower tangent of p_i with respect to E, or*

(ii) *e is non-flippable in $T^*(E)$.*

Proof. First let us consider the case when $e = (p_i, p_j)$ is either the upper or lower tangent of p_i with respect to E. Clearly, e is also either the upper or lower tangent of p_i with respect to $E-e$. Since $T^*(F-e)$ contains the upper and lower tangents for every $p \in P$ by the definition of Construction 1, we obtain $e \in T^*(E-e)$. Thus, $T^*(E-e) = T^*(E)$ holds by Lemma 3.2.

Next let us consider the case when e is non-flippable in $T^*(F)$. Suppose e is either the upper or lower tangent of p_i with respect to E. Then the statement follows from (i). Hence, let us assume that e is neither the upper nor lower tangent with respect to E. We will show $e \in T^*(E-e)$.

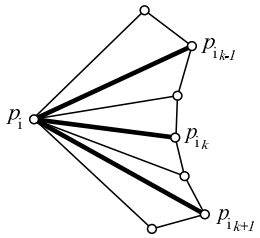


Figure 5: A part of $T^*(E)$, where the bold edges represent E.

According to the way of Construction 1 for $T^*(E)$, the right endpoints of $\delta_E(p_i) \cup \{(p_i, p_i^{\mathrm{up}}), (p_i, p_i^{\mathrm{low}})\}$ are denoted by $p_{i_0}, p_{i_1}, \ldots, p_{i_m}$ in clockwise ordering around p_i, where (p_i, p_i^{up}) and $(p_i, p_i^{\mathrm{low}})$ are the upper and lower tangents of p_i with respect to E. Consider m convex hulls H_k of $P_{i+1} \cap C_k$, for $k = 0, \ldots, m-1$, bounded by the consecutive edges, (p_i, p_{i_k}) and $(p_i, p_{i_{k+1}})$, and then consider the convex chain as the boundary of the convex hull H_k which consists of the sequence of the points $p \in P$ satisfying $p = (p_i, p) \cap H_k$.

Since e is in E (and more precisely $e \in \delta_E(p_i)$) and e is neither the upper nor lower tangent, there exists a subscript k' with $k' \neq 0, m$ for which $e = (p_i, p_{i_{k'}})$ holds. Therefore, since e is non-flippable in $T^*(E)$, combining two convex chains, one from $p_{i_{k'-1}}$ to $p_{i_{k'}}$ and the other from $p_{i_{k'}}$ to $p_{i_{k'+1}}$, we obtain a single convex chain from $p_{i_{k'-1}}$ to $p_{i_{k'+1}}$ (see Fig. 5). This implies that we obtain a convex hull H of the point set P_{i+1} inside the cone bounded by two consecutive edges $(p_i, p_{i_{k'-1}})$ and $(p_i, p_{i_{k'+1}})$ of $(\delta_E(p_i) \setminus \{e\}) \cup \{(p_i, p_i^{\mathrm{up}}), (p_i, p_i^{\mathrm{low}})\}$, (implying that H will be obtained by Construction 1 for $T^*(E-e)$), in which $p_{i_{k'}} = (p_i, p_{i_{k'}}) \cap H$ holds. Hence, e is chosen as the edge of $T^*(E-e)$ in Construction 1, and consequently $T^*(E-e) = T^*(E)$ holds by Lemma 3.2. □

Lemma 3.4. *Let E be a non-crossing edge set and let $e = (p_i, p_j) \in E$. Then, for every $p \in \{p_1, \ldots, p_{i-1}\}$, $\delta_{T^*(E-e)}(p) = \delta_{T^*(E)}(p)$ holds.*

Proof. Let $p \in \{p_1, \ldots, p_{i-1}\}$. Suppose the upper tangent of p with respect to E and that of p with respect to $E - e$ are distinct. Then, $e = (p_i, p_j)$ must properly intersect the upper tangent of p with respect to $E - e$. However, from Observation 2.1, we obtain $p_i < p$, which contradicts $p \in \{p_1, \ldots, p_{i-1}\}$. A similar argument applies to the lower tangent of p. Therefore the upper and lower tangents of p do not change between E and $E - e$. Thus, for every $p \in \{p_1, \ldots, p_{i-1}\}$, Construction 1 for $T^*(E)$ and that for $T^*(E - e)$ produce the same sequence of the convex hulls H_k around p because the upper and lower tangents does not change and $\delta_E(p) = \delta_{(E-e)}(p)$ holds. This implies $\delta_{T^*(E)}(p) = \delta_{T^*(E-e)}(p)$. □

Lemma 3.5. *Let E be a non-crossing edge set. Then, for $e = (p_i, p_j) \in E$, $T^*(E - e) \neq T^*(E)$ holds if e is flippable in $T^*(E)$ and is neither the upper nor lower tangent of p_i with respect to E. Moreover, $T^*(E - e)$ is lexicographically larger than $T^*(E)$.*

Proof. From Lemma 3.4, $\delta_{T^*(E-e)}(p) = \delta_{T^*(E)}(p)$ holds for every $p \in \{p_1, \ldots, p_{i-1}\}$. Hence, let us show that $\delta_{T^*(E-e)}(p_i)$ is a proper subset of $\delta_{T^*(E)}(p_i)$, since if so, the edge list of $T^*(E - e)$ is clearly lexicographically larger than that of $T^*(E)$.

Consider again constructing $T^*(E)$ around p_i by Construction 1. Let $p_{i_0}, p_{i_1}, \ldots, p_{i_m}$ be the right endpoints of $\delta_E(p_i) \cup \{(p_i, p_i^{\mathrm{up}}), (p_i, p_i^{\mathrm{low}})\}$, where (p_i, p_i^{up}) and $(p_i, p_i^{\mathrm{low}})$ are the upper and lower tangents of p_i with respect to E, and let C_k and H_k be the corresponding cone with the apex p_i and the convex hull of $P_{i+1} \cap C_k$ for $0 \leq k \leq m - 1$. From $(p_i, p_j) \in \delta_E(p_i)$, there exists a subscript k' for which $(p_i, p_{i_{k'}}) = (p_i, p_j)$ holds. Moreover, since (p_i, p_j) is neither the upper nor

lower tangent, we have $k' \neq 0, m$. Hence, two convex hulls $H_{k'-1}$ (bounded by $(p_i, p_{i_{k'-1}})$ and $(p_i, p_{i_{k'}})$) and $H_{k'}$ (bounded by $(p_i, p_{i_{k'}})$ and $(p_i, p_{i_{k'+1}})$) are well defined.

Next let us consider $T^*(E-e)$ around p_i by Construction 1. Then, it can be easily observed that the difference between the construction for $T^*(E-e)$ and that for $T^*(E)$ around p_i occurs only in the region bounded by $(p_i, p_{i_{k'-1}})$ and $(p_i, p_{i_{k'+1}})$, which is a cone with the apex p_i, that is $C_{k'-1} \cup C_{k'}$. Let H' be the convex hull of P_{i+1} inside of $C_{k'-1} \cup C_{k'}$. Then, notice $(H_{k'-1} \cup H_{k'}) \subseteq H'$, and notice also that for any $p \in P_{i+1} \cap H'$ either $p \in H_{k'-1}$ or $p \in H_{k'}$ holds. Hence, for any $p \in P_{i+1}$ with $p = (p_i, p) \cap H'$, it holds that either $p = (p_i, p) \cap H_{k'-1}$ or $p = (p_i, p) \cap H_{k'}$, which implies $\delta_{T^*(E-e)}(p_i) \subseteq \delta_{T^*(E)}(p_i)$.

Notice that $p_j (= p_{i_{k'}}) \neq (p_i, p_j) \cap H'$ holds because (p_i, p_j) is flippable in $T^*(E)$. This implies $(p_i, p_j) \notin \delta_{T^*(E-e)}(p_i)$, and hence $\delta_{T^*(E-e)}(p_i) \subsetneq \delta_{T^*(E)}(p_i)$ holds. □

4 Constrained Non-crossing Spanning Trees

Let F be a non-crossing edge set on P, and we assume that F is a forest. In this section we shall show that the collection of the F-constrained non-crossing spanning trees on P, denoted by $\mathcal{CST}$, is connected by $O(n^2)$ flips. A *remove-add flip* for an F-constrained non-crossing spanning tree ST is defined as an operation that removes one edge e_1 with $e_1 \notin F$ from ST and then inserts the other edge $e_2 \in K_n \setminus ST$ into $ST - e_1$ to produce a new F-constrained non-crossing spanning tree $ST - e_1 + e_2$. The *lexicographical order* of the non-crossing spanning trees is similarly defined based on the edge list as that of the triangulations.

Define $\mathcal{CST}^* \subseteq \mathcal{CST}$ as $\mathcal{CST}^* = \{ST \in \mathcal{CST} \mid ST \subset T^*(F)\}$, and ST^* as the F-constrained non-crossing spanning tree consisting of the lexicographically largest edge list among $\mathcal{CST}^*$. Let us first focus on the non-crossing spanning trees contained in $\mathcal{CST}^*$.

Lemma 4.1. *Every non-crossing spanning tree of $\mathcal{CST}^*$ can be transformed into ST^* by at most $n-1$ remove-add flips, each increasing the lexicographical order.*

Proof. Let us consider $ST \in \mathcal{CST}^*$. Let $\{e_1, \ldots, e_{n-1}\}$ and $\{e_1^*, \ldots, e_{n-1}^*\}$ be the edge lists of ST and ST^*, respectively, with $e_1 \prec \cdots \prec e_{n-1}$ and $e_1^* \prec \cdots \prec e_{n-1}^*$. We remove from ST the smallest edge $e_i \in ST \setminus ST^*$ with respect to the edge ordering $\prec$. Note that i is the smallest label such that $e_i \neq e_i^*$. Moreover, we have $e_i \prec e_i^*$ because ST^* has the lexicographically largest edge list among $\mathcal{CST}^*$. When removing e_i from ST, the resulting graph $ST - e_i$ consists of two connected components. Since ST^* is connected, there always exists an edge e_j^* in $ST^* \setminus ST$ which spans the two connected components of $ST - e_i$, and thus $ST - e_i + e_j^*$ is a spanning tree. (This fact is just the rephrase of the basis exchange property of the graphic matroid, see e.g. [23].) Notice that the planarity is maintained since both ST and ST^* are subsets of $T^*(F)$. Moreover, by the definition of the label i, $ST - e_i + e_j^*$ has a lexicographically larger edge list than that of ST. Repeating this process in at most $n-1$ times, we eventually obtain ST^*. □

We associate the ST-constrained lexicographically largest triangulation $T^*(ST)$ with each F-constrained non-crossing spanning tree ST. The sequence of improving flips in the associated triangulation $T^*(ST)$ plays a crucial role when characterizing the remove-add flips of ST, where the edge flip in $T^*(ST)$ should be defined not over $T^*(ST) \setminus ST$ but over $T^*(ST) \setminus F$ because the fixed edges are only those of F, and hence an edge of $ST \setminus F$ may be flippable in $T^*(ST)$. In particular, we shall consider each $T^*(ST)$ as one of *F-constrained* triangulations in the subsequent discussions and the *critical vertex* of $T^*(ST)$ is similarly defined as the vertex having the smallest label i for which $\delta_{T^*(ST)}(p_i) \neq \delta_{T^*(F)}(p_i)$. Fig. 6 shows an example of $T^*(ST)$ whose critical vertex is 4.

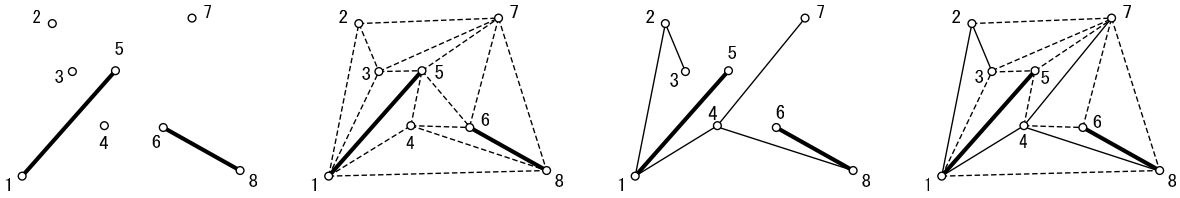


Figure 6: (a) F, (b) $T^*(F)$, (c) ST, and (d) $T^*(ST)$, where the bold edges represent F and the dotted edges represent the edges added to obtain $T^*(F)$ and $T^*(ST)$, respectively.

It is clear from the definition of Construction 1 that the newly added edges to obtain $T^*(ST)$ from ST are not flippable in $T^*(ST)$ except for the upper and lower tangents. Thus, the next observation follows:

Observation 4.2. *Any edge e of $T^*(ST) \setminus ST$ is either* (i) *non-flippable in $T^*(ST)$, or* (ii) *either the upper or lower tangent of the left endpoint of e with respect to ST.*

In addition, we show the following lemma that characterizes the improving flippable edges in $T^*(ST)$.

Lemma 4.3. *An edge $e \in T^*(ST) \setminus F$ is improving flippable in $T^*(ST)$ if and only if e is flippable in $T^*(ST)$ and e is neither the upper nor lower tangent of the left endpoint of e with respect to ST.*

Proof. Let $e = (p_i, p_j)$. ("if"-part:) Consider two triangle faces (p_i, p_j, v) and (p_i, p_j, w) incident to (p_i, p_j) in $T^*(ST)$ with $v, w \in P$. Then, since (p_i, p_j) is neither the upper nor lower tangent with respect to ST, Construction 1 implies $p_i < v$ and $p_i < w$. Hence, flipping (p_i, p_j) increases the lexicographical ordering of $T^*(ST)$.

("only if"-part:) We easily verify that none of the upper and lower tangents of any point is improving flippable as follows. Let us consider an upper tangent, say (p_i, p_i^{up}) (the other case is similarly proved.) When it is flippable, there exists a triangle face $(p_i, p_i^{\mathrm{up}}, v)$ incident to (p_i, p_i^{up}) in $T^*(ST)$ with $v \in P$ and $\Delta(p_i, p_i^{\mathrm{up}}, v) > 0$. The definition of the upper tangent tells us $v < p_i$, and hence flipping (p_i, p_i^{up}) does not increase the lexicographical ordering of the triangulation. □

The following lemma is an immediate corollary of Observation 4.2 and Lemma 4.3.

Lemma 4.4. *Any improving flippable edge in $T^*(ST)$ is contained in $ST \setminus F$.*

We derive the following lemma from Lemma 2.5 and Lemma 4.4:

Lemma 4.5. *Let $ST \in \mathcal{CST} \setminus \mathcal{CST}^*$ and p_c be the critical vertex of $T^*(ST)$. Then, the smallest improving flippable edge in $T^*(ST)$ with respect to the edge ordering $\prec$ always exists among $\delta_{ST}(p_c) \setminus F$.*

Proof. Notice that $\delta_{T^*(ST)}(p_i) = \delta_{T^*(F)}(p_i)$ holds for every $p_i \in \{p_1, \ldots, p_{c-1}\}$ because p_c is the smallest labeled vertex for which $\delta_{T^*(ST)}(p_i) \neq \delta_{T^*(F)}(p_i)$. Hence, every edge $e \in \delta_{T^*(ST)}(p_i)$ with $p_i \in \{p_1, \ldots, p_{c-1}\}$ is incident to the same triangle faces in $T^*(ST)$ as those in $T^*(F)$. Thus, e is not improving flippable in $T^*(ST)$ since otherwise it is also improving flippable in $T^*(F)$, contradicting the fact that $T^*(F)$ is the lexicographically largest F-constrained triangulation. Therefore, there is no improving flippable edge among $\delta_{T^*(ST)}(p_i)$ for all $p_i \in \{p_1, \ldots, p_{c-1}\}$.

Since $T^*(ST) \neq T^*(F)$, there exists at least one improving flippable edge incident to p_c by Lemma 2.5. Moreover, it is an edge of $\delta_{ST}(p_c) \setminus F$ by Lemma 4.4. Thus, the statement is proved. □

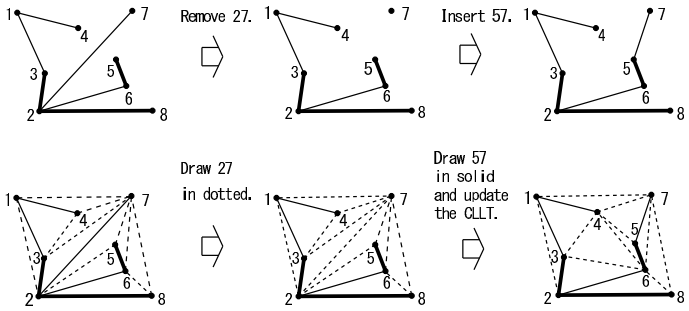


Figure 7: An example of the parent function for $ST \notin \mathcal{CST}^*$, where $F = \{(2,3),(2,8),(5,6)\}$. Removing $(2,7)$ and adding $(5,7)$ we obtain a new spanning tree having the lexicographically larger edge list than the previous one.

Now we are ready to show the existence of a sequence of remove-add flips.

Lemma 4.6. *Every F-constrained non-crossing spanning tree $ST \notin \mathcal{CST}^*$ can be transformed to an F-constrained non-crossing spanning tree contained in $\mathcal{CST}^*$ by $O(n^2)$ remove-add flips, each increasing the lexicographical order.*

Proof. Let p_c be the critical vertex of $T^*(ST)$. From Lemma 4.5 there exists an edge $(p_c, p_{c^*}) \in ST \setminus F$ which is improving flippable in $T^*(ST)$. Let us consider the point $w \in P$ incident to both p_{c^*} and p_c in $T^*(ST)$ such that (p_c, p_{c^*}, w) forms a triangle face of $T^*(ST)$ with $\Delta(p_c, p_{c^*}, w) < 0$. Then, we have $p_c < w$ since (p_c, p_{c^*}) is neither the upper nor lower tangent of p_c by Lemma 4.3. When removing (p_c, p_{c^*}) from ST, the set of vertices of $ST - (p_c, p_{c^*})$ is partitioned into two connected components, where p_{c^*} and p_c belong to the different connected components, and w can belong to only one of them. Therefore, adding one of (p_c, w) or (w, p_{c^*}), we obtain a new F-constrained non-crossing spanning tree ST'. Notice that $(p_c, p_{c^*}) \prec (p_c, w)$ and $(p_c, p_{c^*}) \prec (w, p_{c^*})$ hold and hence ST' has a lexicographically larger edge list than that of ST. Moreover, $T^*(ST')$ is lexicographically larger than $T^*(ST)$ because we remove the improving flippable edge (p_c, p_{c^*}) of $T^*(ST)$ and thus Lemma 3.5 can be applied. Therefore, by repeating this procedure $O(n^2)$ times, the underlying triangulation becomes $T^*(F)$, and then the corresponding F-constrained non-crossing spanning tree is one of $\mathcal{CST}^*$. □

Due to Lemmas 4.1 and 4.6, ST^* can be considered as the *F-constrained lexicographically largest non-crossing spanning tree*, and as a result we obtain the following theorem:

Theorem 4.7. *Let P be a set of n points in the plane. Every F-constrained non-crossing spanning tree on P can be transformed into the F-constrained lexicographically largest non-crossing spanning tree on P by $O(n^2)$ remove-add flips, each increasing the lexicographical order.*

5 Enumerating Constrained Non-crossing Spanning Trees

Let ST^* be the F-constrained lexicographically largest non-crossing spanning tree as defined above. For an edge set E, $\max\{e \in E\}$ and $\min\{e \in E\}$ denote the largest and smallest elements in E with respect to the edge ordering $\prec$, respectively. We define the following *parent function* $f : \mathcal{CST} \setminus \{ST^*\} \to \mathcal{CST}$ based on the results in the previous section. (Recall that the smallest improving flippable edge of $T^*(ST)$ always exists in $\delta_{ST}(p_c) \setminus F$ from Lemma 4.5.)

Definition 5.1. (*Parent function*) *Let $ST \in \mathcal{CST}$ with $ST \neq ST^*$, and p_c be the critical vertex of $T^*(ST)$. Then, $ST' = ST - e_1 + e_2$ is the parent of ST, where*

Algorithm Enumerating F-constrained non-crossing spanning trees.

1: $ST^* :=$ F-constrained lexicographically largest non-crossing spanning tree;
2: $ST' := ST^*$; $i, j := 0$; Output(ST');
3: **repeat**
4: **while** $i \leq |ST'|$ **do**
5: **repeat** $\{i := i + 1; e_{\rm rem} := \mathsf{elist}_{ST'}(i);\}$ **until** $e_{\rm rem} \in F$;
6: **while** $j \leq |K_n|$ **do**
7: **repeat** $\{j := j + 1; e_{\rm add} := \mathsf{elist}_{K_n}(j);\}$ **until** $e_{\rm add} \in ST'$;
8: **if** $ST' - e_{\rm rem} + e_{\rm add} \in \mathcal{CST}$ **then**
9: **if** $f_1(ST' - e_{\rm rem} + e_{\rm add}) = ST'$ **or** $f_2(ST' - e_{\rm rem} + e_{\rm add}) = ST'$ **then**
10: $ST' := ST' - e_{\rm rem} + e_{\rm add}$; $i, j := 0$; Output(ST');
11: **go to** line 4;
12: **end if**
13: **end if**
14: **end while**
15: **end while**
16: **if** $ST' \neq ST^*$ **then**
17: $ST := ST'$;
18: **if** $ST \in \mathcal{CST}^*$ **then** $ST' := f_1(ST)$; **else** $ST' := f_2(ST)$;
19: determine integer pair (i, j) such that $ST' - \mathsf{elist}_{ST'}(i) + \mathsf{elist}_{K_n}(j) = ST$;
20: $i := i - 1$;
21: **end if**
22: **until** $ST' = ST^*$ **and** $i = |ST'|$ **and** $j = |K_n|$;

Figure 8: Algorithm for enumerating F-constrained non-crossing spanning trees.

Case 1: $ST \in \mathcal{CST}^*$,
• $e_1 = \min\{e \mid e \in ST \setminus ST^*\}$, *and* $e_2 = \max\{e \in ST^* \setminus ST \mid ST - e_1 + e \in \mathcal{CST}\}$,
Case 2: $ST \notin \mathcal{CST}^*$,
• $e_1 = (p_c, p_{c^*})$ *is the smallest improving flippable edge in* $T^*(ST)$ *with respect to* $\prec$, *and* e_2 *is either* (p_c, w) *or* (w, p_{c^*}) *such that* $ST - e_1 + e_2 \in \mathcal{CST}$, *where* w *is the vertex of the triangle face* (p_c, p_{c^*}, w) *of* $T^*(ST)$ *with* $\Delta(p_c, p_{c^*}, w) < 0$.

Fig. 7 shows how the parent function works in Case 2. From Lemmas 4.1 and 4.6, these parent-child relations are well defined, and they form the search tree of $\mathcal{CST}$ explained in Introduction. To simplify the notations, we denote the parent function depending on Cases 1 and 2 by $f_1 : \mathcal{CST}^* \setminus \{ST^*\} \to \mathcal{CST}^*$ and $f_2 : \mathcal{CST} \setminus \mathcal{CST}^* \to \mathcal{CST}$, respectively.

Let $\mathsf{elist}_{ST'}$ and elist_{K_n} be the lexicographically ordered edge lists of an F-constrained non-crossing spanning tree ST' and the complete graph K_n on P, and let $\mathsf{elist}_{ST'}(i)$ and $\mathsf{elist}_{K_n}(i)$ be their i-th elements, respectively. Then, based on the algorithm in [7, 8], we describe our algorithm in Fig. 8. The parent function needs $O(n + T_{\rm CLLT})$ time for each execution, where $T_{\rm CLLT}$ denotes the time to calculate $T^*(ST' - e_{\rm rem} + e_{\rm add})$. The while-loop from lines 4 to 15 has $|ST'| \cdot |K_n|$ iterations which requires $O(n^3(n + T_{\rm CLLT}))$ time if simply checking the line 9. We will improve it to $O(n^2)$ time in the next section.

6 Detailed Analysis of the Algorithm

We devote this section to proving the following theorem:

Theorem 6.1. *Let* P *be a set of* n *points in the plane. The set of all the* F*-constrained non-crossing spanning trees on* P *can be enumerated in* $O(n^2)$ *time per output using* $O(n^2)$ *space.*

Let ST' be an F-constrained non-crossing spanning tree on P. Our goal is to enumerate in $O(n^2)$ time all the edge pairs $(e_{\rm rem}, e_{\rm add}) \in ST' \setminus F \times K_n \setminus ST'$ such that $ST = ST' - e_{\rm rem} + e_{\rm add}$ is a *child* of ST'. More precisely, we will show the algorithm for enumerating all $(e_{\rm rem}, e_{\rm add})$ satisfying either $f_1(ST' - e_{\rm rem} + e_{\rm add}) = ST'$ or $f_2(ST' - e_{\rm rem} + e_{\rm add}) = ST'$ among $(ST' \setminus F) \times (K_n \setminus ST')$ in $O(n^2)$ time. The number of candidate pairs $(e_{\rm rem}, e_{\rm add})$ seems to be $O(n^3)$, but in fact it can be reduced to $O(n^2)$. It is because that two edges, e_1 and e_2, involved in the removed-add flip in Case 1 are contained in $T^*(F)$, while e_1 and e_2 in Case 2 are sharing one endpoint. In the followings we will separately consider this enumeration problem for Case 1 and Case 2 (of the parent function).

6.1 Checking $f_1(ST' - e_{\rm rem} + e_{\rm add}) = ST'$

First we show the following lemma which contributes to the efficient checking of whether $f_1(ST' - e_{\rm rem} + e_{\rm add}) = ST'$ holds or not. (The proof can be done in the same manner as that of Lemma 3 of [9].)

Lemma 6.2. *Let ST and ST' be two distinct elements of $\mathcal{CST}^*$ for which $ST = ST' - e_{\rm rem} + e_{\rm add}$ for $e_{\rm rem} \in ST' \setminus F$ and $e_{\rm add} \in T^*(F) \setminus ST'$. Then, $f_1(ST) = ST'$ holds if and only if $(e_{\rm rem}, e_{\rm add})$ satisfies the following conditions:*
(a) $e_{\rm rem} \in ST^*$,
(b) $e_{\rm add} \in T^*(F) \setminus ST^*$,
(c) $e_{\rm rem} \succ \max\{e \in ST^* \setminus ST' \mid ST' - e_{\rm rem} + e \in \mathcal{CST}\}$,
(d) $e_{\rm add} \prec \min\{e \mid e \in ST' \setminus ST^*\}$.

Proof. ("only if"-part.) Since $f_1(ST) = ST'$ holds, we have $e_{\rm rem} = e_2$ and $e_{\rm add} = e_1$, where e_1 and e_2 are the edges defined in Case 1 of Definition 5.1. Hence, by Definition 5.1, $e_{\rm add} = e_1 \in ST \setminus ST^*$ holds, and moreover $ST \in \mathcal{CST}^*$ implies $ST \subset T^*(F)$. Hence we obtain $e_{\rm add} \in T^*(F) \setminus ST^*$, which is (b). Similarly, $e_{\rm rem} = e_2 \in ST^* \setminus ST$ implies (a).

Note that $ST' - e_{\rm rem} = (ST - e_1 + e_2) - e_{\rm rem} = ST - e_1$ holds. Hence, we can see

$$\begin{aligned}
e_{\rm rem} &= \max\{e \in ST^* \setminus ST \mid ST - e_1 + e \in \mathcal{CST}\} && \text{(by Definition 1)}\\
&= \max\{e \in ST^* \setminus (ST' - e_{\rm rem} + e_{\rm add}) \mid ST' - e_{\rm rem} + e \in \mathcal{CST}\} && \text{(by } ST - e_1 = ST' - e_{\rm rem}\text{)}\\
&= \max\{e \in ST^* \setminus (ST' - e_{\rm rem}) \mid ST' - e_{\rm rem} + e \in \mathcal{CST}\} && \text{(by } e_{\rm add} \notin ST^*\text{)}\\
&\succ \max\{e \in ST^* \setminus ST' \mid ST' - e_{\rm rem} + e \in \mathcal{CST}\} && \text{(by } e_{\rm rem} \in ST^* \cap ST'\text{)},
\end{aligned}$$

which implies (c). Similarly, we have

$$\begin{aligned}
e_{\rm add} &= \min\{e \mid e \in ST \setminus ST^*\} && \text{(by Definition 1)}\\
&= \min\{e \mid e \in (ST' - e_{\rm rem} + e_{\rm add}) \setminus ST^*\}\\
&\prec \min\{e \mid e \in ST' \setminus ST^*\} && \text{(by } e_{\rm add} \notin ST^* \cup ST' \text{ and } e_{\rm rem} \in ST^*\text{)},
\end{aligned}$$

which implies (d).

("if"-part.) Since $e_{\rm rem} \in ST^*$ by (a), (d) implies

$$\begin{aligned}
e_{\rm add} &\prec \min\{e \mid e \in ST' \setminus ST^*\}\\
&= \min\{e \mid e \in (ST + e_{\rm rem} - e_{\rm add}) \setminus ST^*\}\\
&= \min\{e \mid e \in (ST - e_{\rm add}) \setminus ST^*\}.
\end{aligned}$$

Thus, $e_{\rm add} = \min\{e \mid e \in ST \setminus ST^*\}$ holds, and hence f_1 chooses $e_{\rm add}$ for the edge e_1 to be deleted from ST according to Definition 5.1. Similarly, we have

$$\begin{aligned} e_{\rm rem} &\succ \max\{e \in ST^* \setminus ST' \mid ST' - e_{\rm rem} + e \in \mathcal{CST}\} && \text{(by (c))} \\ &= \max\{e \in ST^* \setminus (ST + e_{\rm rem} - e_{\rm add}) \mid ST - e_1 + e \in \mathcal{CST}\} && \text{(by } ST' - e_{\rm rem} = ST - e_1\text{)} \\ &= \max\{e \in ST^* \setminus (ST + e_{\rm rem}) \mid ST - e_1 + e \in \mathcal{CST}\} && \text{(by (b)).} \end{aligned}$$

Thus, since $e_{\rm rem} \in ST^* \setminus ST$, we obtain $e_{\rm rem} = \max\{e \in ST^* \setminus ST \mid ST - e_1 + e \in \mathcal{CST}\}$. Therefore, f_1 chooses $e_{\rm rem}$ for the edge to be added, and $f_1(ST)$ returns ST'. □

Note that Lemma 6.2 states that no child of ST' exists in the search tree if there is no $(e_{\rm rem}, e_{\rm add}) \in ST' \setminus F \times T^*(F) \setminus ST'$ satisfying all the conditions of Lemma 6.2. For example, there is no child of ST' if $ST^* \cap ST' = \emptyset$ by the condition (a).

Lemma 6.3. *Given $ST' \in \mathcal{CST}^*$, all pairs $(e_{\rm rem}, e_{\rm add}) \in ST' \setminus F \times T^*(F) \setminus ST'$ satisfying the conditions of Lemma 6.2 such that $ST' - e_{\rm rem} + e_{\rm add} \in \mathcal{CST}$ can be enumerated in $O(n^2)$ time.*

Proof. We assume that ST^* and $T^*(F)$ are pre-computed in the preprocessing phase before the enumeration, and the edge sets of $ST' \setminus F$, $ST^* \setminus F$ and $T^*(F) \setminus F$ are maintained in lexicographically ordered edge lists, respectively. For the edges to be added, using linear time, the algorithm computes the edge $e' = \min\{e \mid e \in ST' \setminus ST^*\}$, and then it computes the edge list of $T^*(F) \setminus (ST^* \cup ST')$ each of whose elements is smaller than e' with respect to the ordering $\prec$. Let us denote this edge list by L. Then, note that every edge $e_{\rm add} \in L$ satisfies the conditions (b) and (d).

Similarly, since $e_{\rm rem}$ must be in $(ST' \cap ST^*) \setminus F$ from the condition (a), the algorithm computes an edge list of $(ST' \cap ST^*) \setminus F$ in $O(n)$ time, and then examine each edge of $(ST' \cap ST^*) \setminus F$ one by one as follows. For each $e_{\rm rem} \in (ST' \cap ST^*) \setminus F$, by taking $O(n)$ time, (i) it checks whether $e_{\rm rem}$ satisfies the condition (c), and (ii) it enumerates all the edges $e_{\rm add}$ among L such that $ST' - e_{\rm rem} + e_{\rm add} \in \mathcal{CST}$. For each vertex v, the algorithm computes which connected component the vertex v belongs to in $ST' - e_{\rm rem}$ by using $O(n)$ time so that it can check in $O(1)$ time whether $ST' - e_{\rm add} + e$ forms a spanning tree for an edge e. Then, the algorithm can compute the edge $e'' = \max\{e \in ST^* \setminus ST' \mid ST' - e_{\rm rem} + e \in \mathcal{CST}\}$ in $O(n)$ time by checking each edge of $ST^* \setminus ST'$ one by one whether it spans the different components of $ST' - e_{\rm rem}$. (Note that all edges in $ST^* \cup ST'$ are non-crossing.) Similarly, it can enumerate in $O(n)$ time all the $e_{\rm add} \in L$ such that $ST' - e_{\rm rem} + e_{\rm add} \in \mathcal{CST}$ since $|L| = O(n)$. Thus, we can obtain the desired edge pairs $(e_{\rm rem}, e_{\rm add})$ in $O(n)$ time for each $e_{\rm rem} \in (ST' \cap ST^*) \setminus F$, and the lemma follows. □

6.2 Checking $f_2(ST' - e_{\rm rem} + e_{\rm add}) = ST'$

Next we will explain how we can efficiently check whether $f_2(ST' - e_{\rm rem} + e_{\rm add}) = ST'$ holds or not. Consider the situation that we first add $e_{\rm add} = (p_x, p_y)$ to ST' and then remove $e_{\rm rem}$ from $ST' + e_{\rm add}$ such that $ST' - e_{\rm rem} + e_{\rm add} \in \mathcal{CST}$. From Definition 5.1, $e_{\rm rem}$ and $e_{\rm add}$ must share exactly one endpoint if $f_2(ST' - e_{\rm rem} + e_{\rm add}) = ST'$ holds. We hence denote by p_z the other endpoint of $e_{\rm rem}$ that is not shared by $e_{\rm add}$. Now let us characterize the edge pair $(e_{\rm rem}, e_{\rm add})$.

Lemma 6.4. *Let ST and ST' be two distinct F-constrained non-crossing spanning trees for which $ST = ST' - e_{\rm rem} + e_{\rm add}$ for $e_{\rm add} = (p_x, p_y) \in K_n \setminus ST'$ and $e_{\rm rem} \in ST' \setminus F$ that is either (p_x, p_z) or (p_y, p_z) for some $p_z \in P$. Then, $f_2(ST) = ST'$ holds if and only if $(e_{\rm rem}, e_{\rm add})$ satisfies the following conditions:*

(A) *the triangle face (p_x, p_y, p_z) exists in $T^*(ST)$ with $\Delta(p_x, p_y, p_z) < 0$.*

(B) *e_{add} is the smallest improving flippable edge in $T^*(ST)$ with respect to the edge ordering $\prec$.*

Proof. The necessary and sufficient condition for $f_2(ST) = ST'$ is that $e_{\mathrm{rem}} = e_2$ and $e_{\mathrm{add}} = e_1$ hold, where e_1 and e_2 are those defined in Case 2 of Definition 5.1. Hence, replacing e_1 and e_2 of Definition 5.1 by e_{add} and e_{rem}, respectively, we obtain the conditions (A) and (B). □

To enumerate all the pairs of $(e_{\mathrm{rem}}, e_{\mathrm{add}})$ satisfying the conditions (A) and (B) of Lemma 6.4, we will check one by one whether each pair satisfies these conditions. However, since $ST' \setminus F = O(n)$ and $K_n \setminus ST' = O(n^2)$, it takes $O(n^3)$ time if we reconstruct $T^*(ST)(= T^*(ST' + e_{\mathrm{add}} - e_{\mathrm{rem}}))$ explicitly from $T^*(ST')$ to check the conditions. In fact Lemma 6.4 does not directly provide an efficient algorithm, and then we will consider a further characterization of each condition, which is described in terms of $T^*(ST' + e_{\mathrm{add}})$ in the subsequent lemmas.

We remark here that $T^*(ST' + e_{\mathrm{add}})$ is not well defined if $ST' + e_{\mathrm{add}}$ is crossing. When first adding e_{add} to ST' in order to obtain a new non-crossing spanning tree $ST' - e_{\mathrm{rem}} + e_{\mathrm{add}}$, there are two situations: e_{add} does not properly intersect any edge of ST' or e_{add} properly intersects one edge e of ST' (in this case e must be removed as e_{rem}). However, as noticed above, e_{add} and e_{rem} share one endpoint for every pair $(e_{\mathrm{rem}}, e_{\mathrm{add}})$ to satisfy $f_2(ST' - e_{\mathrm{rem}} + e_{\mathrm{add}}) = ST'$, and consequently e_{add} does not properly intersect e_{rem}. Hence, we may restrict our attention to e_{add} such that $ST' + e_{\mathrm{add}}$ is non-crossing. We first consider the condition (A) of Lemma 6.4:

Lemma 6.5. *The condition (A) of Lemma 6.4 holds, i.e., the triangle face (p_x, p_y, p_z) exists in $T^*(ST' - e_{\mathrm{rem}} + e_{\mathrm{add}})$ with $\Delta(p_x, p_y, p_z) < 0$ if and only if*

(A-a) *e_{rem} is either non-flippable in $T^*(ST' + e_{\mathrm{add}})$, or the upper or lower tangent of the left endpoint of e_{rem} with respect to $ST' + e_{\mathrm{add}}$, and*

(A-b) *the triangle face (p_x, p_y, p_z) exists in $T^*(ST' + e_{\mathrm{add}})$ with $\Delta(p_x, p_y, p_z) < 0$.*

Proof. ("only if"-part) Since the triangle face (p_x, p_y, p_z) exists in $T^*(ST' - e_{\mathrm{rem}} + e_{\mathrm{add}})$, we have $e_{\mathrm{rem}} \in T^*(ST' - e_{\mathrm{rem}} + e_{\mathrm{add}})$, and hence, by Lemma 3.2, we have $T^*(ST' - e_{\mathrm{rem}} + e_{\mathrm{add}}) = T^*((ST' - e_{\mathrm{rem}} + e_{\mathrm{add}}) + e_{\mathrm{rem}}) = T^*(ST' + e_{\mathrm{add}})$. Thus, (A-b) holds. Moreover, from Observation 4.2, e_{rem} is either non-flippable in $T^*(ST' - e_{\mathrm{rem}} + e_{\mathrm{add}})$, or the upper or lower tangent of the left endpoint of e_{rem} with respect to $ST' - e_{\mathrm{rem}} + e_{\mathrm{add}}$. Therefore, e_{rem} is either non-flippable in $T^*(ST' + e_{\mathrm{add}})$, or the upper or lower tangent with respect to $ST' + e_{\mathrm{add}}$, implying (A-a).

("if"-part) From Lemma 3.3 with the condition (A-a), we have $T^*(ST' + e_{\mathrm{add}}) = T^*(ST' + e_{\mathrm{add}} - e_{\mathrm{rem}})$. Therefore, from (A-b), $T^*(ST' - e_{\mathrm{rem}} + e_{\mathrm{add}})$ contains the triangle face (p_x, p_y, p_z) with $\Delta(p_x, p_y, p_z) < 0$. □

Next let us characterize the condition (B) of Lemma 6.4 by the following two lemmas.

Lemma 6.6. *Let ST' be a non-crossing spanning tree, and $e_{\mathrm{add}} \in K_n \setminus ST'$ be an edge such that $ST' + e_{\mathrm{add}}$ is non-crossing. Then, the following two facts hold:*

(1) *e_{add} is improving flippable in $T^*(ST' + e_{\mathrm{add}})$ if and only if $e_{\mathrm{add}} \notin T^*(ST')$.*

(2) *Every edge $e \in ST'$ with $e \prec e_{\mathrm{add}}$ that is not improving flippable in $T^*(ST')$ remains non-improving flippable in $T^*(ST' + e_{\mathrm{add}})$.*

Proof. Let $e_{\mathrm{add}} = (p_x, p_y)$. First let us show (1). Suppose $e_{\mathrm{add}} \in T^*(ST')$. Then, $e_{\mathrm{add}} \in T^*(ST') \setminus ST'$ implies that e_{add} is not improving flippable in $T^*(ST')$ from Lemma 4.4. Moreover, from Lemma 3.2 and Observation 4.2, we have $T^*(ST') = T^*(ST' + e_{\mathrm{add}})$, and thus "only-if" part of (1) holds. To prove "if"-part of (1) let us assume $e_{\mathrm{add}} \notin T^*(ST')$. Since $T^*(ST')$ contains both

the upper and lower tangents of p_x by the definition of Construction 1, e_{add} is neither the upper nor lower tangent of p_x with respect to ST'. Moreover, since the addition of $e_{\mathrm{add}} = (p_x, p_y)$ does not affect the visibility of p_x, e_{add} is also neither the upper nor lower tangent of p_x with respect to $ST' + e_{\mathrm{add}}$. Thus, e_{add} is improving flippable in $T^*(ST' + e_{\mathrm{add}})$ if e_{add} is flippable in $T^*(ST' + e_{\mathrm{add}})$ by Lemma 4.3. Suppose, for a contradiction, that e_{add} is not flippable in $T^*(ST' + e_{\mathrm{add}})$. Then, we obtain $T^*(ST' + e_{\mathrm{add}}) = T^*(ST')$ from Lemma 3.3, and thus $e_{\mathrm{add}} \in T^*(ST' + e_{\mathrm{add}}) = T^*(ST')$, which contradicts the assumption $e_{\mathrm{add}} \notin T^*(ST')$.

Next let us consider (2). We assume $e_{\mathrm{add}} \notin T^*(ST')$ since otherwise $T^*(ST' + e_{\mathrm{add}}) = T^*(ST')$ holds from Lemma 3.2 and hence the statement clearly holds. Let us denote the edges of $\delta_{ST'}(p_x) \cup \{(p_x, p_x^{\mathrm{up}}), (p_x, p_x^{\mathrm{low}})\}$ by $(p_x, p_{x_1}), \ldots, (p_x, p_{x_m})$ in the clockwise order around p_x. Note that $(p_x, p_x^{\mathrm{up}}) \prec e_{\mathrm{add}} \prec (p_x, p_x^{\mathrm{low}})$ holds because, if $e_{\mathrm{add}} \prec (p_x, p_x^{\mathrm{up}})$, e_{add} intersects some edge of ST' from the definition of the upper tangent, which contradicts that $ST' + e_{\mathrm{add}}$ is non-crossing. Similarly, $(p_x, p_x^{\mathrm{low}}) \prec e_{\mathrm{add}}$ cannot happen. Moreover, $e_{\mathrm{add}} \notin T^*(ST')$ implies that e_{add} is neither the upper nor lower tangent of p_x. Thus, there exists the subscript k with $0 \leq k < m$ satisfying $(p_x, p_{x_k}) \prec e_{\mathrm{add}} = (p_x, p_y) \prec (p_x, p_{x_{k+1}})$.

Let e be an edge in $\{e \in ST' \setminus \{(p_x, p_{x_k})\} \mid e \prec e_{\mathrm{add}}\}$. Then, we claim that two triangle faces Δ_1 and Δ_2 incident to e in $T^*(ST')$ do not change in $T^*(ST' + e_{\mathrm{add}})$. To see this, we have two cases depending on the position of the left endpoint p of e. When $p \in \{p_1, \ldots, p_{x-1}\}$, from Lemma 3.4, we have $\delta_{T^*(ST')}(p) = \delta_{T^*(ST' + e_{\mathrm{add}})}(p)$. Thus, Δ_1 and Δ_2 remain in $T^*(ST' + e_{\mathrm{add}})$. When $p = p_x$, recall that every edge of $T^*(ST')$ that does not properly intersect e_{add} remains in $T^*(ST' + e_{\mathrm{add}})$ by Lemma 3.1. It is obvious that e_{add} does not properly intersect any edge of Δ_1 and Δ_2. Thus, every edge $e \in ST'$ with $e \prec e_{\mathrm{add}}$ that is not improving flippable in $T^*(ST')$ remains non-improving flippable in $T^*(ST' + e_{\mathrm{add}})$ except for (p_x, p_{x_k}).

The proof is completed by showing that (p_x, p_{x_k}) is not improving flippable in $T^*(ST' + e_{\mathrm{add}})$ when (p_x, p_{x_k}) is not in $T^*(ST')$. Let C_k be the cone with the apex p_x obtained in Construction 1 for $T^*(ST')$ around p_x, which is bounded by two consecutive edges (p_x, p_{x_k}) and $(p_x, p_{x_{k+1}})$, and let H_k be the convex hull of $P_{x+1} \cap C_k$. Since $(p_x, p_{x_k}) \prec e_{\mathrm{add}} = (p_x, p_y) \prec (p_x, p_{x_{k+1}})$ and $e_{\mathrm{add}} \notin T^*(ST')$, H_k completely contains p_y as shown in Fig. 9(a).

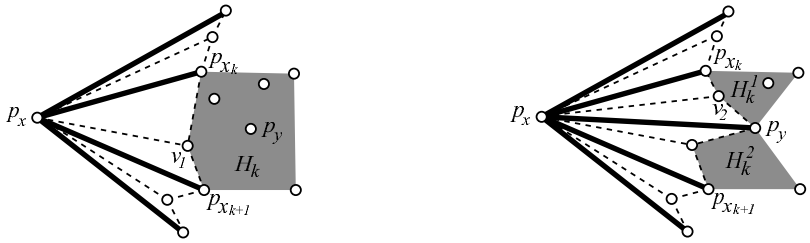
When constructing $T^*(ST' + e_{\mathrm{add}})$ around p_x, the convex hull H_k is divided into two convex hulls, denoted by H_k^1 and H_k^2: one is bounded by (p_x, p_{x_k}) and e_{add} and the other is bounded by e_{add} and $(p_x, p_{x_{k+1}})$ (see Fig. 9(b)). Note that $H_k^1 \subset H_k$ and $H_k^2 \subset H_k$ hold. Let us consider the case when (p_x, p_{x_k}) is not improving flippable in $T^*(ST')$. Then, by Lemma 4.3, (p_x, p_{x_k}) is either (i) non-flippable in $T^*(ST')$, or (ii) the upper tangent of p_x with respect to ST'.

(i) When (p_x, p_{x_k}) is non-flippable in $T^*(ST')$, let us denote by (p_x, p_{x_k}, v_1) a triangle face of $T^*(ST')$ incident to (p_x, p_{x_k}) with $\Delta(p_x, p_{x_k}, v_1) < 0$. Similarly, let (p_x, p_{x_k}, v_2) be that of $T^*(ST' + e_{\mathrm{add}})$ (see Fig. 9). Notice that $v_1 \in H_k$ and $v_2 \in H_k^1$, and hence $v_2 \in H_k$ holds from $H_k^1 \subset H_k$. Therefore, the angle $\angle p_x p_{x_k} v_1$ around p_{x_k} is smaller than or equal to the angle $\angle p_x p_{x_k} v_2$, and thus (p_x, p_{x_k}) is again non-flippable in $T^*(ST' + e_{\mathrm{add}})$. (Note that the triangle face incident to (p_x, p_{x_k}) in the opposite side does not change when adding e_{add}.)

(ii) When (p_x, p_{x_k}) is the upper tangent of p_x with respect to ST', it remains as the upper tangent of p_x with respect to $ST' + e_{\mathrm{add}}$ because $e_{\mathrm{add}} = (p_x, p_y)$ does not affect the visibility from p_x. Hence, (p_x, p_{x_k}) is not improving flippable edge in $T^*(ST' + e_{\mathrm{add}})$ by Lemma 4.3. □

Lemma 6.7. *Let p_c be the critical vertex of $T^*(ST')$, and let $p_{c_1}, \ldots, p_{c_m}$ be the right endpoints of $\delta_{ST'}(p_c) \cup \{(p_c, p_c^{\mathrm{up}}), (p_c, p_c^{\mathrm{low}})\}$ around p_c in clockwise order. Let $(p_c, p_{c_{k^*}})$ be the smallest improving flippable edge in $T^*(ST')$ with respect to $\prec$. Then, $e_{\mathrm{add}} = (p_x, p_y) \in K_n \setminus ST'$ is the smallest improving flippable edge in $T^*(ST' + e_{\mathrm{add}})$ with respect to $\prec$ if and only if*

(B-a) *$e_{\mathrm{add}} \notin T^*(ST')$, and*



(a) (b)

Figure 9: The figures (a) and (b) show the parts of $T^*(ST')$ and $T^*(ST' + e_{\text{add}})$ around $e_{\text{add}} = (p_x, p_y)$, respectively. The bold edges represent the constrained edges, and the dotted edges represent the other edges appeared in each triangulation. The edge (p_x, p_{x_k}) is non-flippable in $T^*(ST' + e_{\text{add}})$ if it is so in $T^*(ST')$.

(B-b) *either* (*i*) $e_{\text{add}} \prec (p_c, p_{c_{k^*}})$ *holds, or* (*ii*) $(p_c, p_{c_{k^*}}) \prec e_{\text{add}} \prec (p_c, p_{c_{k^*+1}})$ *holds and* $(p_c, p_{c_{k^*}})$ *is not improving flippable in* $T^*(ST' + e_{\text{add}})$.

Proof. ("only if"-part) From Lemma 6.6, (B-a) holds. Suppose (B-b) does not hold. Then, either one of the following two cases occurs: (1) $(p_c, p_{c_{k^*+1}}) \prec e_{\text{add}}$ or (2) $(p_c, p_{c_{k^*}}) \prec e_{\text{add}}$ and $(p_c, p_{c_{k^*}})$ is improving flippable in $T^*(ST' + e_{\text{add}})$. It is obvious that Case (2) cannot occur since otherwise the existence of the improving flippable edge $(p_c, p_{c_{k^*}})$, which is smaller than e_{add}, contradicts that e_{add} is the smallest one among the improving flippable edges in $T^*(ST' + e_{\text{add}})$. Suppose Case (1) occurs. Then, $(p_c, p_{c_{k^*}})$ is incident to the same two triangle faces in $T^*(ST' + e_{\text{add}})$ as those in $T^*(ST')$ from Lemma 3.1, and hence $(p_c, p_{c_{k^*}})$, which is smaller than e_{add}, remains improving flippable in $T^*(ST' + e_{\text{add}})$, again. This is a contradiction and thus (B-b) holds.

("if"-part) From Lemma 6.6 and (B-a), e_{add} is improving flippable in $T^*(ST' + e_{\text{add}})$. Let us show that e_{add} is actually the smallest one in $T^*(ST' + e_{\text{add}})$. Note that every $e \in T^*(ST' + e_{\text{add}}) \setminus (ST' + e_{\text{add}})$ cannot be improving flippable in $T^*(ST' + e_{\text{add}})$ by Lemma 4.4.

Let us consider (B-b). If (B-b)(i) holds (i.e. $e_{\text{add}} \prec (p_c, p_{c_{k^*}})$), there exists no improving flippable edge among $\{e \in T^*(ST') | e \prec e_{\text{add}}\}$ because $(p_c, p_{c_{k^*}})$ is the smallest improving flippable edge in $T^*(ST')$. Hence, e_{add} is the smallest improving flippable in $T^*(ST' + e_{\text{add}})$ since every edge $e \in ST'$ with $e \prec e_{\text{add}}$ that is not improving flippable in $T^*(ST')$ remains non-improving flippable in $T^*(ST' + e_{\text{add}})$ by Lemma 6.6.

If (B-b)(ii) holds, it holds that there exists no improving flippable edge in $\{e \in T^*(ST') | e \prec e_{\text{add}}\}$ except for $(p_c, p_{c_{k^*}})$ because $(p_c, p_{c_{k^*}})$ is the smallest improving flippable edge in $T^*(ST')$. By (B-b)(ii), $(p_c, p_{c_{k^*}})$ is not improving flippable in $T^*(ST' + e_{\text{add}})$. Thus, e_{add} is the smallest improving flippable in $T^*(ST' + e_{\text{add}})$ by Lemma 6.6, again. Therefore, in both cases, there exists no improving flippable edge among $\{e \in T^*(ST' + e_{\text{add}}) \mid e \prec e_{\text{add}}\}$. □

Notice that Lemma 6.7 considers $T^*(ST' + e_{\text{add}})$, but not $T^*(ST' - e_{\text{rem}} + e_{\text{add}})$, which implies that e_{add} may not be the smallest improving flippable edge in $T^*(ST' - e_{\text{rem}} + e_{\text{add}})$ (i.e., e_{add} violates the condition (B) of Lemma 6.4), even if e_{add} satisfies both conditions (B-a) and (B-b) of Lemma 6.7. However, in the situation that the condition (A-a) of Lemma 6.5 holds, we have $T^*(ST' - e_{\text{rem}} + e_{\text{add}}) = T^*(ST' + e_{\text{add}})$ by Lemma 3.3, and hence the condition (B) of Lemma 6.4 holds if and only if e_{add} is the smallest improving flippable edge in $T^*(ST' + e_{\text{add}})$. As a consequence, combining Lemmas 6.4, 6.5 and 6.7, it follows that $ST' - e_{\text{rem}} + e_{\text{add}}$ is a child of ST' with respect to f_2 if and only if

- $ST' - e_{\text{rem}} + e_{\text{add}}$ forms a non-crossing spanning tree, and

- $(e_{\rm rem}, e_{\rm add})$ satisfies all the conditions (A-a), (A-b), (B-a) and (B-b) of Lemmas 6.5 and 6.7.

We first show that, with $O(n^2)$ time preprocessing and $O(n^2)$ space, we can check whether $ST' - e_{\rm rem} + e_{\rm add}$ forms a spanning tree in $O(1)$ time for any $(e_{\rm rem}, e_{\rm add})$. Then, we shall provide a way to obtain the set of edges $e_{\rm add} = (p_x, p_y)$ among $\delta_{K_n}(p_x)$ such that $ST' + e_{\rm add}$ is non-crossing. This process takes $O(d(p_x)n)$ time for each $p_x \in P$, where $d(p_x)$ denotes the degree of p_x in ST'. By using these methods, we shall provide an algorithm for enumerating all the pairs $(e_{\rm rem}, e_{\rm add})$ with $e_{\rm add} \in \delta_{K_n}(p_x)$ satisfying the condition (A-b) and $ST' - e_{\rm rem} + e_{\rm add} \in \mathcal{CST}$. This takes $O(d(p_x)n)$ time for each $p_x \in P$. Thus, the total time of this process for all $p_x \in P$ becomes $O(n^2)$. Finally we will show how to check whether the obtained pairs $(e_{\rm rem}, e_{\rm add})$ satisfy all the other conditions, (A-a), (B-a) and (B-b) in $O(n^2)$ time. As a result, we will obtain the following lemma:

Lemma 6.8. *Given $ST' \in \mathcal{CST}$, all pairs $(e_{\rm rem}, e_{\rm add}) \in (ST' \setminus F) \times (K_n \setminus ST')$ for which $f_2(ST' - e_{\rm rem} + e_{\rm add}) = ST'$ can be enumerated in $O(n^2)$ time with $O(n^2)$ space.*

Proof. The proof is divided into six parts since it is long.

(i)Checking whether $ST' - e_{\rm rem} + e_{\rm add}$ is a spanning tree. We can check, with $O(n^2)$ preprocessing time and $O(n^2)$ space, whether $ST' - e_{\rm rem} + e_{\rm add}$ forms a spanning tree in $O(1)$ time for every $(e_{\rm rem}, e_{\rm add})$ as follows. The algorithm computes which connected component every vertex belongs to when removing $e_{\rm rem}$ from ST' by using $O(n)$ time for each $e_{\rm rem} \in ST'$, and it retains this information for every $e_{\rm rem} \in ST'$ so that it can check in $O(1)$ time whether $e_{\rm add}$ spans the different connected components of $ST' - e_{\rm rem}$. The preprocessing takes $O(n^2)$ time and the space requires $O(n^2)$ as all information can be stored in the $|ST'| \times |V|$ matrix.

(ii)Finding $e_{\rm add}$ such that $ST' + e_{\rm add}$ is non-crossing. To find $e_{\rm add}$ which properly intersects no edge of ST', we shall show an efficient way to compute the set of points of $P \setminus \{p\}$ visible from a point $p \in P$. For a simple polygon $\mathcal{P}$ and a vertex $v \in \mathcal{P}$, a *visibility polygon* of v with respect to $\mathcal{P}$ is defined to be $\mathcal{VP}_{\mathcal{P}}(v) = \{p \in \mathbf{R}^2 \mid$ a line segment (v, p) is in $\mathcal{P}\}$. The following fact is known:

Fact 6.9. ([20, 21]) *Let $\mathcal{P}$ be a simple polygon. Then, the visibility polygon of a vertex $v \in \mathcal{P}$ with respect to $\mathcal{P}$ can be computed in linear time.*

On the other hand, in general case, it takes $\Theta(n \log n)$ time to compute the visibility polygon (region), $\mathcal{VP}_S(v) = \{p \in \mathbf{R}^2 \mid p$ is visible from v with respect to $S\}$, for a point p and a set of line segments S [6]. However, in the case of the line segments consisting of a non-crossing spanning tree, computing the visibility polygon (region) can be performed in almost linear time as shown in the following lemma:

Lemma 6.10. *Let ST' be a non-crossing spanning tree on a point set P. Then, the visibility polygon (region) of a point $p \in P$ with respect to the edge set of ST' can be found in $O(d(p)n)$ time, where $d(p)$ is the degree of p in ST'.*

Proof. Let R be an axis-parallel rectangle enclosing ST', and let r be a ray emanating from p_1 to the left side of R. Let us find the visibility polygon of p inside R. We can view the problem of finding the visibility polygon of p with respect to ST' as the one of finding the visibility polygon with respect to $\mathcal{P}$, where $\mathcal{P}$ is the simple polygon obtained by tracing ST', R and r as shown in Fig. 10. Each point p encounters $d(p)$ times during the trace, which produces $d(p)$ vertices of $\mathcal{P}$ that are associated with p. Let us denote these vertices in the order of the tracing by $v_1, \ldots, v_d$ as shown in Fig. 10, where $d = d(p)$. Then, the visibility polygon of p inside R is $\bigcup_i \mathcal{VP}_{\mathcal{P}}(v_i)$. Each

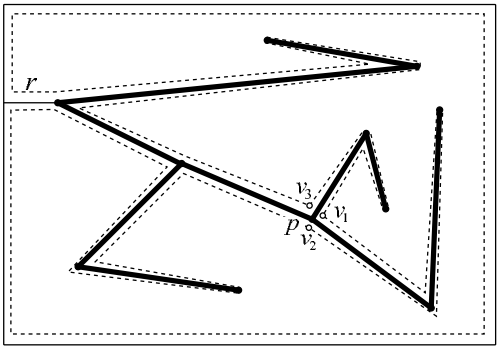


Figure 10: A non-crossing spanning tree ST' contained in a large rectangle R, where the bold edges represent those of ST'. The dotted simple polygon $\mathcal{P}$ is obtained by tracing ST', R and r. The point p is encountered three times, which produces three vertices v_1, v_2 and v_3 of $\mathcal{P}$.

of $\mathcal{VP}_{\mathcal{P}}(v_i)$ can be computed in $O(n)$ time by the algorithm of Fact 6.9, and taking the union can be done in linear time because each $\mathcal{VP}_{\mathcal{P}}(v_i)$ intersects only $\mathcal{VP}_{\mathcal{P}}(v_{i-1})$ and $\mathcal{VP}_{\mathcal{P}}(v_{i+1})$ on a line segment incident to p. □

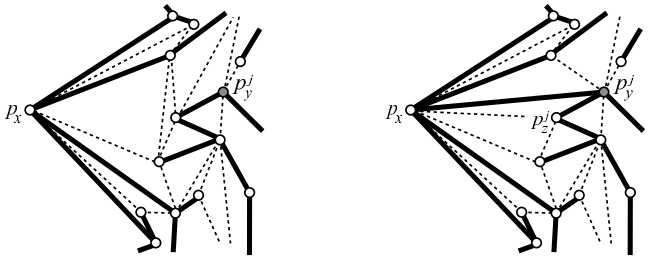
(iii)Finding pairs $(e_{\rm rem}, e_{\rm add})$ satisfying the condition (A-b). We assume that a flag representing whether $e \in ST \setminus F$ or not is attached to each edge e of $T^*(ST')$. Also we assume that the algorithm can check whether $ST' - e_{\rm rem} + e_{\rm add}$ is a spanning tree in $O(1)$ time for any $(e_{\rm rem}, e_{\rm add})$. For every $p_x \in P$ we show an algorithm for enumerating all pairs $(e_{\rm rem}, e_{\rm add})$ with $e_{\rm add} \in \delta_{K_n}(p_x)$ satisfying the condition (A-b) and $ST' - e_{\rm rem} + e_{\rm add} \in \mathcal{CST}$ in $O(\deg(p_x)n)$ time.

We first compute the visibility polygon (region) of p_x with respect to ST' by using $O(d(p_x)n)$ time algorithm described in Lemma 6.10. Since the visibility polygon of p_x is star-shaped with the kernel containing p_x, we can obtain all the vertices of P that are visible from p_x with respect to ST' in clockwise ordering around p_x by tracing the visibility polygon. Denote these points lying on the right side of p_x by $p_y^1, p_y^2, \cdots, p_y^{\bar{j}}$ in the clockwise ordering around p_x, and store the edges (p_x, p_y^j) of $1 \le j \le \bar{j}$ in the list, denoted by L.

The algorithm checks one by one for every element $e_{\rm add} = (p_x, p_y^j)$ of L whether there exists an appropriate edge $e_{\rm rem}$ to be removed such that $(e_{\rm rem}, e_{\rm add})$ satisfies (A-b). Namely, it first inserts a new constrained edge $e_{\rm add} = (p_x, p_y^j)$ into $T^*(ST')$ and then tries to find an appropriate $e_{\rm rem}$ using $T^*(ST' + e_{\rm add})$, (but does not construct the whole $T^*(ST' + e_{\rm add})$ explicitly), for every $e_{\rm add} \in L$.

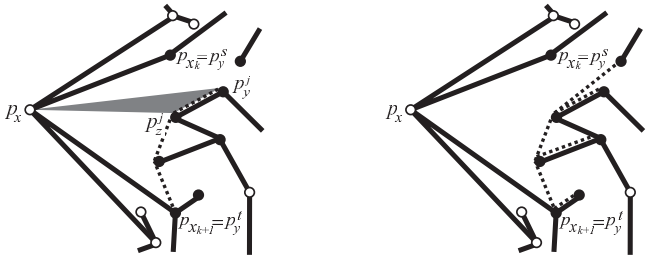
Let (p_x, p_y^j, p_z^j) be the triangle face incident to $e_{\rm add} = (p_x, p_y^j)$ in $T^*(ST' + e_{\rm add})$ in the lower side, i.e., $\Delta(p_x, p_y^j, p_z^j) < 0$. Then, when fixing $e_{\rm add} = (p_x, p_y^j) \in L$, the condition (A-b) restricts the possibility of $e_{\rm rem}$ to only two edges of $T^*(ST' + e_{\rm add})$, i.e., only (p_x, p_z^j) and (p_z^j, p_y^j) may be chosen as $e_{\rm rem}$ (see Fig. 11). The algorithm hence picks up $e_{\rm rem} \in \{(p_x, p_z^j), (p_z^j, p_y^j)\} \cap ST'$ such that $ST' - e_{\rm rem} + e_{\rm add}$ is a spanning tree. (Also, this implies that, if $ST' - e_{\rm rem} + e_{\rm add}$ is not a spanning tree for any $e_{\rm rem} \in \{(p_x, p_z^j), (p_z^j, p_y^j)\}$, there exists no child of ST' for $e_{\rm add} = (p_x, p_y^j)$.) As the algorithm can check in $O(1)$ time whether $ST' - e_{\rm rem} + e_{\rm add}$ is a spanning tree, it can find all (but at most two) $e_{\rm rem}$ such that $ST' - e_{\rm rem} + e_{\rm add}$ is a non-crossing spanning tree and $(e_{\rm rem}, e_{\rm add})$ satisfies (A-b) for each $e_{\rm add}$ in constant time if the algorithm knows the triangle face (p_x, p_y^j, p_z^j). Thus, to find $(e_{\rm add}, e_{\rm rem})$ satisfying the condition (A-b), it is sufficient to calculate only the triangle face (p_x, p_y^j, p_z^j) without calculating the whole $T^*(ST' + e_{\rm add})$. We will show below how to obtain (p_x, p_y^j, p_z^j) for all $(p_x, p_y^j) \in L$ in $O(n)$ time.

Let $p_{x_0}, p_{x_1}, \ldots, p_{x_m}$ be the right endpoints of $\delta_{ST'}(p_x) \cup \{(p_x, p_x^{\rm up}), (p_x, p_x^{\rm low})\}$ in the clock-



(a) (b)

Figure 11: Figures (a) and (b) illustrate $T^*(ST')$ and $T^*(ST' + e_{\rm add})$ around p_x, respectively, where the dotted edges represent those added to be triangulated. In this case, only (p_z^j, p_y^j) may be chosen as the edge $e_{\rm rem}$ to be removed by the condition (A-b).

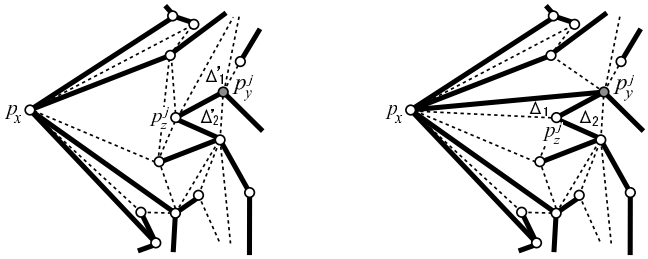


(a) (b)

Figure 12: The figures for explaining the computation of the triangle faces (p_x, p_y^j, p_z^j), where the bold edges represent those of ST' around p_x. The black vertices represent $p_y^s, p_y^{s+1}, \ldots, p_y^{t-1}, p_y^t$ in clockwise ordering around p_x, which are visible from p_x. Notice that the dotted convex chain from p_y^t to p_y^j of Figure (a) is supposed to be obtained in Construction 1 for $T^*(ST' + (p_x, p_y^j))$, and hence the shaded triangle region represents (p_x, p_y^j, p_z^j) in $T^*(ST' + (p_x, p_y^j))$. Figure (b) illustrates the set of such convex chains for all j with $s < j < t$.

wise ordering around p_x, where $(p_x, p_x^{\rm up})$ and $(p_x, p_x^{\rm low})$ are the upper and lower tangents of p_x with respect to ST'. Let us consider the case when p_y^j is contained in the cone C_k bounded by (p_x, p_{x_k}) and $(p_x, p_{x_{k+1}})$. From the definition of the visibility, there exist the superscripts s and t with $1 \leq s \leq t \leq \bar{j}$ such that $p_{x_k} = p_y^s$ and $p_{x_{k+1}} = p_y^t$ among $p_y^1, p_y^2, \cdots, p_y^{\bar{j}}$, and the points $p_y^s, p_y^{s+1}, \cdots, p_y^{t-1}, p_y^t$ are contained in C_k (see Fig. 12(a)). When inserting the new constrained edge (p_x, p_y^j) of $s < j < t$ into $T^*(ST')$, the desired triangle face incident to (p_x, p_y^j) in the lower side can be found in constant time if the algorithm can compute the convex chain from $p_{x_{k+1}} (= p_y^t)$ to p_y^j, which is a boundary of the convex hull bounded by (p_x, p_y^j) and $(p_x, p_{x_{k+1}})$ in $T^*(ST' + (p_x, p_y^j))$ (see Fig. 12(a)). Our algorithm efficiently computes this convex chain connecting between $p_{x_{k+1}}$ and p_y^j for every j with $s < j < t$ by tracing the vertices p_y^j in the ordering of $p_y^t, p_y^{t-1}, \cdots, p_y^{s+1}, p_y^s$ (see Fig. 12(b)).

This can be done by performing Graham scan algorithm [14] (not in the order of the coordinates as usual but in the ordering of $p_y^t, p_y^{t-1}, \cdots, p_y^{s+1}, p_y^s$). In fact, the process of Graham scan will maintain the desired convex chain. When it encounters a new point p_y^j during the scan, it examines the top point p_y^a on the stack and the next one p_y^b. If p_x and p_y^a are in the distinct sides of the line passing through p_y^j and p_y^b, then it pops p_y^a. Continue this process until it obtains three vertices p_y^j, $p_y^{a'}$ and $p_y^{b'}$ such that p_x and $p_y^{a'}$ are in the same side of the line through p_y^j and $p_y^{b'}$, (or until the



(a) (b)

Figure 13: Figures (a) and (b) illustrate $T^*(ST')$ and $T^*(ST'+e_{\rm add})$ around p_x, respectively, where the dotted edges represent those added to be triangulated. Δ_i' and Δ_i for $i=1,2$ are triangles incident to (p_z^j, p_y^j) in $T^*(ST')$ and $T^*(ST'+e_{\rm add})$, respectively.

stack contains only one vertex $p_y^{a'}$). It obtains the desired convex chain from p_y^t to p_y^j. This process can be performed in time proportional to $t-s+1$ (the number of p_y^j with $s \le j \le t$). Therefore, by performing this process inside every cone C_k with $0 \le k < m$, the algorithm computes the desired triangle faces (p_x, p_y^j, p_z^j) for all $e_{\rm add} = (p_x, p_y^j) \in L$ in $O(n)$ time. Thus, all the pairs $(e_{\rm rem}, e_{\rm add})$ with $e_{\rm add} = (p_x, p_y^j)$ satisfying the condition (A-b) can be computed in $O(d(p_x)n)$ for each $p_x \in P$.

(iv)Checking the condition (A-a). Next, let us consider how to verify whether the candidate edge pairs $(e_{\rm rem}, e_{\rm add})$ obtained in the above process **(iii)** satisfy the condition (A-a). Notice that the algorithm can check the condition (A-a) in constant time for each $e_{\rm rem}$ if it can obtain the two triangle faces of $T^*(ST'+e_{\rm add})$ incident to $e_{\rm rem}$. Let us denote these two triangle faces of $T^*(ST'+e_{\rm add})$ by Δ_1 and Δ_2 (see Fig. 13(b)).

Recall that, for a candidate pair $(e_{\rm rem}, e_{\rm add})$ with $e_{\rm add} = (p_x, p_y^j)$ calculated in the above process **(iii)**, $e_{\rm rem}$ is either (p_x, p_z^j) or (p_z^j, p_y^j). Hence, one of the two triangles, say Δ_1, is (p_x, p_y^j, p_z^j).

Let us consider how to obtain the other triangle Δ_2. Let Δ_1' and Δ_2' be two triangle faces of $T^*(ST')$ incident to $e_{\rm rem}$. (If $e_{\rm rem}$ is incident to only one triangle face in $T^*(ST')$, then it lies on the boundary of the convex hull of P and $e_{\rm rem}$ is always the upper or lower tangent of the left endpoint of $e_{\rm rem}$. This implies $e_{\rm rem}$ satisfies the condition (A-a).) It is obvious that $e_{\rm add}$ can intersect at most one of Δ_1' and Δ_2', say Δ_1', and hence Δ_2' still exists in $T^*(ST'+e_{\rm add})$ by Lemma 3.1 (see Fig. 13). Clearly, we have $\Delta_2 = \Delta_2'$, and thus the algorithm can check the condition (A-a) in constant time for each $e_{\rm rem}$ by using two triangle faces, $\Delta_1 = (p_x, p_y^j, p_z^j)$ that is already obtained in the process of (A-b) and $\Delta_2 = \Delta_2'$ that already exists in $T^*(ST')$.

(v)Checking the condition (B-a). The condition (B-a) can be easily verified by just avoiding the output of the edge pairs $(e_{\rm rem}, e_{\rm add})$ such that $e_{\rm add} = (p_x, p_y^j) \in T^*(ST')$ during the above process.

(vi)Checking the condition (B-b). Let us explain how to check whether the candidate edge pairs satisfy the condition (B-b). Let $(p_c, p_{c_{k^*}})$ be the smallest improving flippable edge in $T^*(ST')$ with respect to $\prec$, which can be computed in $O(n)$ time for a given $T^*(ST')$ by checking the edges of ST' one by one. As defined in Lemma 6.7, let $(p_c, p_{c_{k^*+1}})$ be the edge of ST' that is next to $(p_c, p_{c_{k^*}})$ with respect to the edge ordering $\prec$ among the edges in ST'. Then it can be checked in constant time whether $e_{\rm add} \prec (p_c, p_{c_{k^*}})$ or not. If not and $(p_c, p_{c_{k^*}}) \prec e_{\rm add} \prec (p_c, p_{c_{k^*+1}})$ holds, the algorithm needs to check whether $(p_c, p_{c_{k^*}})$ is improving flippable or not in $T^*(ST'+e_{\rm add})$. This can be done in $O(1)$ time if we have the two triangle faces incident to $(p_c, p_{c_{k^*}})$. Applying the exactly same

method as was done in **(iii)**, the algorithm updates the triangle faces incident to $(p_c, p_{c_{k^*}})$ in the lower side when inserting $e_{\rm add}$ without calculating whole $T^*(ST'+e_{\rm add})$; maintaining a convex chain between $p_{c_{k^*}}$ and p_y^j when inserting (p_x, p_y^j) one by one among $(p_c, p_{c_{k^*}}) \prec (p_x, p_y^j) \prec (p_c, p_{c_{k^*+1}})$. That is to say, the condition (B-b) can be checked in $O(n)$ time in total. □

As a result we complete the proof of Theorem 6.1 from Lemmas 6.3 and 6.8.

7 Concluding Remarks

We have presented algorithms for enumerating all the edge-constrained triangulations and all the edge-constrained non-crossing geometric spanning trees based on the edge-constrained lexicographically largest triangulation. We have also provided several geometric properties of the edge-constrained lexicographically largest triangulation in Sections 2 and 3. In our recent paper [22], using the edge-constrained lexicographically largest triangulation as well as the results of Section 3, we have newly revealed combinatorial properties that relate the non-crossing geometric graphs and the edge-constrained lexicographically largest triangulation on a point set. Based on the properties, we have proposed a general framework for efficiently enumerating a large class of non-crossing geometric graphs such as plane straight-line graphs, non-crossing spanning connected graphs, (unconstrained) non-crossing spanning trees, non-crossing minimally rigid graphs, non-crossing matchings, non-crossing blue-and-red matchings and etc.

We note in passing that the techniques proposed in this paper can also be used to defined a local operation and an efficient enumeration algorithm for the edge-constrained non-crossing connected spanning graphs whose unconstrained case was considered in [2]. An open problem, which is of considerably practical importance, is to efficiently generate all the non-crossing spanning trees on P that do not contain a given edge set. This problem is challenging because it is known that determining if a geometric graph contains a non-crossing spanning tree is NP-complete [19].

Acknowledgment

We would like to thank Professor David Avis and Professor Ileana Streinu for many research discussions about the enumeration of non-crossing geometric graphs. The first author is supported by the project *New Horizons in Computing*, Grant-in-Aid for Scientific Research on Priority Areas, MEXT Japan. The second author is supported by Grant-in-Aid for JSPS Research Fellowships for Young Scientists.

References

[1] O. Aichholzer, F. Aurenhammer, C. Huemer and H. Krasser. Transforming spanning trees and pseudo-triangulations. *Inf. Process. Lett.*, 97(1):19–22, 2006.

[2] O. Aichholzer, F. Aurenhammer, C. Huemer, and B. Vogtenhuber. Gray code enumeration of plane straight-line graphs. *Graphs and Combinatorics*, 23(5):467–479, 2007.

[3] O. Aichholzer, F. Aurenhammer and F. Hurtado. Sequences of spanning trees and a fixed tree theorem. *Comput. Geom.*, 21(1-2):3–20, 2002.

[4] O. Aichholzer, T. Hackl, C. Huemer, F. Hurtado, H. Krasser, and B. Vogtenhuber. On the number of plane geometric graphs. *Graphs and Combinatorics*, 23[Suppl]:67–84, 2007.

[5] O. Aichholzer and K. Reinhardt. A quadratic distance bound on sliding between crossing-free spanning trees. *Comput. Geom.*, 37:155–161, 2007.

[6] T. Asano, S. K. Ghosh and T. Shermer. Visibility in the plane. In J.-R Sack and J. Urrutia eds., *Handbook of Computational Geometry*, Elsevier, Chapter 19, pp. 829–876, 2000.

[7] D. Avis and K. Fukuda. A pivoting algorithm for convex hulls and vertex enumeration of arrangements and polyhedra. *Discrete and Computational Geometry*, 8:295–313, 1992.

[8] D. Avis and K. Fukuda. Reverse search for enumeration. *Discrete Applied Mathematics*, 65(1-3):21–46, March 1996.

[9] D. Avis, N. Katoh, M. Ohsaki, I. Streinu and S. Tanigawa. Enumerating constrained non-crossing minimally rigid frameworks. *Discrete and Computational Geometry*, 40(1):31–46, 2008.

[10] S. Bereg. Enumerating pseudo-triangulations in the plane. *Comput. Geom. Theory Appl.*, 30(3):207–222, 2005.

[11] M. Bern and D. Eppstein. Mesh generation and optimal triangulation. *Computing in Euclidean Geometry, 2nd Edition, Du and Hwang eds.*, 23–90, 1992.

[12] S. Bespamyatnikh. An efficient algorithm for enumeration of triangulations. *Comput. Geom. Theory Appl.*, 23(3):271–279, 2002.

[13] H. Brönnimann, L. Kettner, M. Pocchiola and J. Snoeyink. Enumerating and counting pseudo-triangulations with the greedy flip algorithm. *SIAM J. Comput.*, 36(3):721–739, 2006.

[14] R. L. Graham. An efficient algorithm for determining the convex hull of a finite planar set. *Inf. Process. Lett.*, 1:132–133, 1972.

[15] M. C. Hernando, M. E. Houle and F. Hurtado. On local transformation of polygons with visibility properties. In *Theoretical Computer Science*, 289(2):919–937, 2002.

[16] C. Hernando, F. Hurtado and M. Noy. Graphs of non-crossing perfect matchings. *Graphs and Combinatorics*, 18(3):517–532, 2002.

[17] M. E. Houle, F. Hurtado and M. Noy and E. Rivera-Campo, Graphs of triangulations and perfect matchings, *Graphs and Combinatorics*, 21(3):325-331, 2005.

[18] F. Hurtado, M. Noy and J. Urrutia. Flipping edges in triangulations. *Discrete & Computational Geometry*, 22(3):333–346, 1999.

[19] K. Jansen and G. J. Woeginger. The complexity of detecting crossingfree configurations in the plane. *BIT* 33(4):580–595, 1993.

[20] B. Joe and R. B. Simpson. Corrections to Lee's visibility polygon algorithm. *BIT*, 27(4):458–473, 1987.

[21] D. T. Lee. Visibility of a simple polygon. *Computer Vision, Graphics, and Image Processing*, 22(2):207–221, 1983.

[22] N. Katoh and S. Tanigawa. Fast enumeration algorithms for non-crossing geometric graphs. *Proc. 24th ACM Symposium on Computational Geometry*, 328–337, 2008.

[23] D. J. A. Welsh. Matroids: fundamental concepts. In *Handbook of Combinatorics Vo.I, R.L.Graham, M.Grötschel, and L.Lovász eds.*, North-Holland, 1995, 481-526.